Masters Thesis TRITA-ICT-EX-2009:11

An Authentication Framework for Nomadic Users in Ubiquitous Computing

Master of Science Thesis
In Electronic System Design

by

NAVEED AHMED

Stockholm, May 2009

Supervisor: Christian Damsgaard Jensen
Examiner: Axel Jantsch

ISBN: 978-91-7415-367-5

Author

Naveed Ahmed
nahm@kth.se
naah@student.dtu.dk
Ph: +46 73 9638409, +45 5290 2953

Supervisor

Christian Damsgaard Jensen
Associate Professor
Department of Informatics and Mathematical Modeling
Room 013, Building 322, Richard Petersens Plads
Technical University of Denmark (DTU)
DK-2800 Lyngby, Denmark
Telephone: +45 4525 3351
Direct line: +45 4525 3724
Fax: +45 4593 0074
http://www2.imm.dtu.dk/~cdj
Email: Christian.Jensen@imm.dtu.dk

Examiner

Axel Jantsch
Professor
Department of Electronic, Communication and Software Systems
School for Information and Communication Technology
Royal Institute of Technology (KTH)
Stockholm, Sweden
Phone: +46 8 790 4124, +46 70 713 7428
Fax: +46 8 751 1793
http://web.it.kth.se/~axel
Email: axel@imit.kth.se

Technical University of Denmark

Informatics and Mathematical Modeling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone: +45 4525 3351
Fax: +45 4588 2673
reception@imm.dtu.dk
www.imm.dtu.dk

Royal Institute of Technology

School of Information and Communication Technology
KTH-Kista, Ingenjörsskolan, Electrum 213,
SE-164 40 KISTA, Stockholm, Sweden
Phone: +46 8 752 14 00
Fax: +46 8 751 15 44
info@ict.kth.se
http://www.kth.se/ict

Acknowledgments

First of all, I would like to thank my supervisor Christian Damsgaard Jensen at the Technical University Denmark (DTU), for all the guidance and motivation that he provided throughout the project. I would also like to express my gratitude for all the arrangements he made for my enrollment and accommodation in DTU. I would also like to thank Patrick Konneman at DTU for providing help from time to time, and the department secretary for her administrative support.

I am very thankful to Prof. Axel Jantsch at the Royal Institute of Technology (KTH), Sweden, for his prompt replies, support and administrative help during the project. I would also like to thank Mr. Wajid H. Minhass at KTH for his constructive feedback on all plans and documents regarding the project.

I owe to my family in Pakistan for their utmost moral support and prays.

At last but not least, a resounding thank to the Higher Education Commission of Pakistan (HEC) and Swedish Institute (SI), Sweden, for funding my living expenses in Sweden and Denmark during the duration of the project.

Abstract

Security and usability are often horn locked and at times, security experts tend to make a system more secure on expense of usability. For nomadic users, however, this approach is highly undesirable and due to poor resultant usability, users start feeling the need to circumvent the security mechanism. Although designed with a high level of security, such a system becomes quite vulnerable to various security threats. One of the best example of this is classic password based authentication mechanism when used by nomadic users. For instance in hospitals, patients' treatment is at top priority. Doctors and nurses are usually in urgency and also often require to authenticate for a couple of times within a single hour, to access sensitive patients' health data. It is also very typical that a single terminal is being shared by many staff members, over a course of time. Moreover, they often need to delegate their duties to other co-workers. In most of these places, people tend to use short and easy passwords, going out with out logout, sharing passwords with colleagues, etc. As a consequence, security of the system has deteriorated considerably in the trade off with usability of the authentication mechanism.

After analyzing these types of usability problems, we have compiled a list of key requirements which must be considered while designing an authentication mechanism for nomadic use. These requirements specify the nature of user interaction in an authentication mechanism and are aimed to improve usability experience as well as the effective security of a system. To meet these requirements, we have proposed a network based authentication framework, called NDAF (Nomadic Delegation and Authentication Framework). This framework supports zero interaction, persistent and multi factor techniques. We have also introduced the concept of delegation at user authentication level in the framework. This is essentially equivalent to giving one's password to another person, but is secure, persistent and accountable. Furthermore, due to distributed network nature, its integration with session migration is trivial.

A prototype of the proposed authentication framework has been developed, which supports persistent and multi-factor authentication without the active intervention of a user. We have used device centric authentication based on RFID tags, which represents a single branch in multi-factor authentication. The client and the server part of the mechanism are present on a single computer. This implemented mechanism also supports multiple simultaneous active sessions and authentication level user delegation. Currently, we have not demonstrated session migration features in our experiment due to time constraints for the project.

We have evaluated the developed mechanism from both usability and security perspective, and have compared it to classic knowledge based authentication. The evaluation shows that by abating usability constraints, an increase in the effective level of security is achieved. Also, it is evident from our experiment that it saves substantial time of nomadic users which otherwise is being drained while authenticating. Thus it can provide users with more job satisfaction and increased level security, which definitely contribute to organizational productivity.

Keywords: Security, Usability, Authentication, Delegation, Nomadic Users, Persistent Authentication, Zero-Interaction, NDAF

Preface

This report documents the Masters Thesis research and development work which is carried out at the Department of Informatics and Mathematical Modeling (IMM) at Technical University Denmark (DTU) in the time period between 1st Dec-2008 and 1st May-2009, as a requirement of Masters in 'System on Chip' Design, for Royal Institute of Technology (KTH), Sweden. The workload of this thesis is 30 ECTS points for the author.

The name of the thesis, encapsulates our core research work which is carried out in the field of Ubiquitous computing. The thesis addresses the usability constraints of nomadic users during authentication process. Nomadic users belong to a class of people, who frequently use, share and authenticate on multiple computing devices embedded in their work environments, e.g. hospitals, emergency rescue service, conference halls, etc. We have presented security risks associated with these nomadic users and have suggested a solution. We have implemented a scaled down version of our proposed solution in form of a prototype. The software code for the prototype can be found on the disk accompanying the report.

Mainly, the thesis work is done in three phases. In the first phase, we have analyzed the interaction between nomadic users and classic knowledge based authentication mechanisms. This phase is winded up by listing usability requirements for nomadic users in authentication mechanisms. In the second phase we have developed the architecture for a framework of authentication, which addresses all requirements gathered in the first phase. We call it Nomadic Delegation and Authentication Framework (NDAF). The third phase is an experiment which consists of implementing and analyzing the authentication mechanism based on our proposed framework.

It is important to note that the primary focus of our research is not on increasing the level of security on which one can design an authentication mechanism. Instead, our intent is to remove those usability constraints which adversely affects the level of effective security achieved in actual use of these mechanisms.

This report is aimed for broad range of readers. However, knowledge in one of the following fields might be helpful to cope the essence of this research: Electronics, Informatics, Computer Security, Wireless and Communication. This report is critically reviewed by supervisors and an external person, the opponent. However, if you find any mistake or some information which needs more elaboration, please contact the author.

Naveed Ahmed Kgs. 2800 Lyngby, Copenhagen, May 1st – 2009

Table of Contents

Abbreviations

ACL Access Control List

API Application Programming Interface

BAN The BAN logic[33] was invented by Burrows, Abadi and Needham and hence got this name. It is the first suggestion to formalize authentication protocols for analysis purpose.

CIA Confidentiality, Integrity and Availability

DAC Discretionary Access Control

EPC Electronic Product Code

EPR Electronic Patient Record

FCC Federal Communications Commission. It is a regulatory body in the United States.

GSM Global System for Mobile communications

IBM International Business Machines corporation

LAN Local Area Network

MAC Mandatory Access Control

NBS National Bureau of Standards

NDAF Nomadic Delegation and Authentication Framework

NIST The National Institute of Standards and Technology

OS Operating System

PAM Pluggable Authentication Module

PIN Personal Identification Number

PC Personal Computer

PDA Personal Digital Assistant

RBAC Role Based Access Control

RFID Radio Frequency Identification

UPC Universal Product Code

WORM Write Once, Read Many. A type of read only memory which can only be programmed once.[6]

CHAPTER - 1

Introduction

Over the past few decades, an exponential growth in computing technology and its immense use in our life has caused a shift in the trend of how we interact with computers. Earlier, in the age of mainframes and desktops, a user had to physically go to a computer for the purpose of information or computing. When technology allowed manufacturing of light weighted devices along with a well connected wireless communication infrastructure, the paradigm of mobile computing emerged. Mobile computing allows people to move freely in a environment while easily carrying computing devices with them, e.g. mobile phone, laptops, PDA, etc. More recently, we have started embedding computing devices in work environments, all over the places where they might be required. This allows users to move freely in a environment with out necessarily carrying any computing devices, as computers are already available at desired locations.

As the use of computers spread to all aspects of human activity, the value of the computing resources themselves and the information stored in computers become apparent. The actual value of access to computers and information stored on them depends on the application and the environment in which the computers are used. For example, data on a stock exchange mainframe may have value in terms of billions of dollars. In military command and control, the data present on a computers may have worth in terms of nation's security. Likewise, medical record present on a hospital's computer, have worth that corresponds to patient's health, personal privacy and better social life. Due to these facts, computer security has become as vital as computer itself.

There are different ways to achieve computer security, such as physical security, encrypted file system, secure programming, etc. However, the most common way to achieve computer security is access control mechanisms, provided by operating systems such as Unix, Linux, Windows and Mac OS. An access control mechanism restricts the access to computer resources in accordance to the security policy. Most of access control mechanisms rely on the identity of a user, which is obtained by an authentication mechanism. In such systems, authentication is a prerequisite which must be performed before invoking the access control mechanism. For instance in Linux, every system file has access rights for owner, group and for all of the system users. The Linux access control mechanism only grants a user access to a file if relevant permission bits are set. However, the security of this access control mechanism depends on the login program which provides the system with a validated user identity. If the login program can be deceived, access control can not do much to prevent unauthorized access.

Similar to the Linux login program, all authentication mechanisms are primarily designed to verify that claimed identity of user is valid or not. Mostly, this decision is binary similar to 'yes' or 'no', but in some state of the art this could also be a probability that corresponds to the trust level for a particular authentication technique. There are multiple ways to authenticate a user, however, knowledge based authentication is widely used and is part of many popular operating systems, for example password based login mechanism in Unix, Linux or Windows. In a login program a user enters his identity, the so called user name, and the corresponding password. If hash value of the typed password matches to the stored value in the system, user is authenticated. If we assume that the user chooses a sufficiently long, random and secret password, then this mechanism is fundamentally secure. Other knowledge based authentication mechanisms like passphrases, PIN code, graphical passwords, puzzles, etc, also work on similar assumptions.

In general, knowledge based authentication is quite suitable and secure for personal or mainframe computing, where there is normally no requirement for frequent user movability across different computing devices. On the other hand, there is a class of users, who need to move freely and frequently within their work environment and we call them nomadic users. Nomadic users access different devices, which are also potentially being shared by other users. Usually nomadic users have their unique sessions or roles which they invoke on particular terminals and typically for a shorter period of time. Thus, they

need to authenticate at each new terminal before using it. One can imagine that the frequency of authentication is relatively high in this type of nomadic use of computing. A typical example of such users is hospital staff, where doctors and nurses frequently move from one place to another and access sensitive patients' data from different locations. If they are using some knowledge based authentication like password based login program then, at the end of a day, a significant portion of their time is consumed in this process.

Authentication for Nomadic Users

As indicated earlier, nomadic users need to access different computers during the day, which means that they regularly login, logout and share computing devices with other users after short intervals. Thus, frequent authentication is a noticeable property for nomadic users. Unfortunately, classic knowledge based authentication mechanisms do not accommodate this unique usability requirement. As we know, in knowledge based authentication a conscious user interaction is required. Thus, if the frequency of this interaction is high, then it consumes a significant portion of the time resulting in poor usability and is also referred as 'Usability Constraints'. This stimulates users to find a way to minimize their interaction with the authentication mechanism. However, in doing so, they often circumvent the security of the computer system.

Let us consider the example of a study report from health sector[34], where nomadic users rely on password based authentication. Since nomadic users need to frequently login, they tend to use short and easy to type passwords. Obviously, these passwords violate basic assumptions of security for password based authentication. Short and easy passwords can easily be cracked or guessed by attackers. Furthermore, due to large number of authentication events, users tend to forget to logout and some times they do not logout on purpose in order to save time when they expect to come back after a few minutes. Similarly, they often share passwords with colleagues, among a group and even give away their passwords so that other people can perform their tasks while they are away (this is the example of how authentication mechanisms are sometimes used to delegate access rights to data and computing resources). Unfortunately, if a password is shared then there is no accountability if some things go wrong.

The poor usability of knowledge based authentication causes two types of problems for typical nomadic users. Firstly, a nomadic user's active involvement in the authentication process consumes considerable proportion of his daily work hours, when all small periods of interactions are added together. Secondly, poor usability poses a risk of numerous security vulnerabilities caused by the user's effort to minimize his interaction with the authentication mechanism. As a result, systems which are designed for secure computer use, not only become less secure but also less usable, due to these usability constraints. Therefore, in real life, we are left with nomadic environments which are less productive and vulnerable to serious security threats.

NDAF: An authentication framework

It is clear from the previous discussion that nomadic users represent an important class of computer users and many more computer users may evolve to this class as our environments are becoming more and more ubiquitous. Conventional knowledge based authentication mechanisms do not consider usability needs of nomadic users very well. As a result, nomadic users are exposed to numerous security threats, causing a considerable reduction in effective level of security. These constraints also contribute towards significant loss of time, job dissatisfaction and a noticeable reduction in the organization's productivity.

To address the usability constraints, we must analyze the human-computer interaction during authentication in the context of nomadic users. This allows us to formulate a list of requirements to address usability constraints. As usability is linked to the security of nomadic user, satisfying these requirements also raises the level of effective security. We suggest that these requirements must be considered as an integral part of the design of an authentication mechanism meant for nomadic use. To

demonstrate how these requirements can be met, we propose an authentication framework called NDAF (Nomadic Delegation and Authentication Framework). This is a network based authentication architecture which supports multi factor, persistent and context based authentication. However, NDAF only addresses the usability aspect of authentication and it is therefore independent of the particular authentication techniques, underlying cryptographic primitives and protocols.

We have instantiated the framework into an authentication mechanism and presented it in the form of a prototype using RFID reader and a desktop computer running Debian Linux. The prototype provides zero-interaction, persistent authentication and delegation services. Although the prototype consists of a single computer, but the authentication could be context based that may activate relevant session at nearby computer for a user. We recognize that users often tell their passwords to friends and co-workers which is a form of delegation at user level. This form of delegation, however, provides no accountability, so we have implemented delegation at user authentication level, which can be considered as equivalent to giving one's password to a colleague, but it is secure and persistent. The revocation of delegation is also very simple and takes effect immediately due to the persistent nature of delegation mechanism.

The analysis of the prototype indicates that the effective level of security has considerably increased. This increase in security is achieved by the prevention of common vulnerabilities present in many nomadic environments. In turn, these vulnerabilities are caused by usability constraints present in use of classic knowledge (e.g. passwords) based authentication mechanisms. The usability of security is one of many causes, responsible for not achieving designed level of security when a system is practically used. Further, we have also analyzed our system from pure security point of view, as it may happen that while addressing usability constraints, one may introduce new vulnerabilities.

The thesis work is summarized in Figure 1. As shown, nomadic users belong to an important class of computer users and authentication mechanisms are vital part of computer security. We have studied the interaction between nomadic users and authentication mechanisms which enables us to discover various security vulnerabilities and usability constraints. Based on these observations, we have identified a number of important usability requirements which should be considered for authentication in nomadic environment. To meet these requirements, we have proposed an authentication framework called NDAF (Nomadic Delegation and Authentication Framework). Further, we have instantiated the frame work in form of a prototype authentication mechanism. And finally, we have analyzed the prototype and compared it to the widely used classic knowledge based authentication.

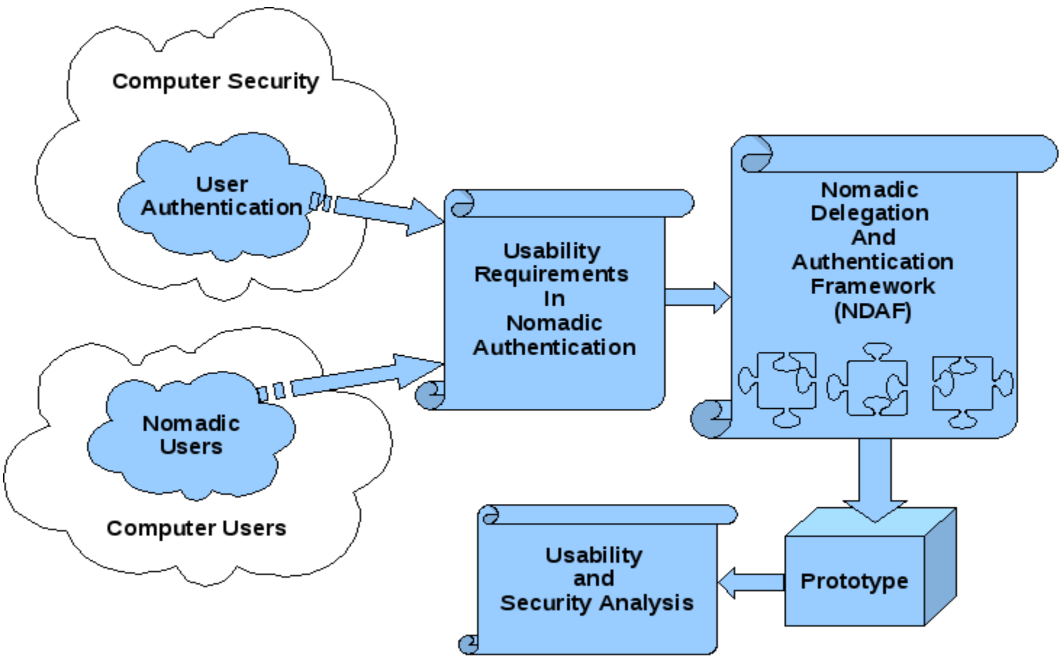

Figure 1: The Thesis Work

We have addressed the usability of authentication system by introducing token based authentication, in an environment where knowledge based authentication is typically used. Moreover, we claim that this may actually improve the security of the overall authentication system, because the improved usability removes an important incentive for users to circumvent the, otherwise, tedious authentication process.

Our analysis of the prototype covers both usability and security perspectives. Usability of the prototype is evaluated empirically, by various user trials, as well as analytically. The result shows it to be very suitable for nomadic users as it helps to save time which other wise is being consumed in the authentication process. A detailed security analysis for our implementation concludes that the prototype has indeed prevented many security vulnerabilities and thus obtains a considerable gain in terms of security over commonly used conventional knowledge based authentication mechanisms. Our implementation is a scaled down version of real life nomadic environments, but it is still sufficient for the proof of concept. It can also be used as a reference to build many security mechanisms to tailor specific needs of a particular nomadic environment.

Structure of the report

Next chapter presents a brief about computer security, nomadic users and their environments. The third chapter highlights usability problems of knowledge based authentication when used in nomadic environments. The fourth chapter consists of our proposed authentication framework which is based on the requirements we induced from observations of the interaction between nomadic users and authentication mechanisms. The sixth chapter is dedicated to the detail of our implemented prototype of the authentication framework. The chapter seven provides security and usability analysis in order to find out the effectiveness of the proposal. The last chapter concludes our work while also providing some future directions.

CHAPTER - 2

Security and Nomadic Users

In this chapter, we introduce some basic concepts of computer security relevant to the authentication process. Further, we describe the evolution of human computer interaction during the past forty years of computing history. Recently, this evolution is dominated by the nomadic use of computing, as our computing infrastructure becomes more networked and ubiquitous. A nomadic computer user moves frequently in the environment and uses different shared computing devices that are part of the infrastructure. A more specific interpretation of a nomadic user and a nomadic environment is presented at the end of this chapter.

Computer Security

Computer security is defined in slightly different ways by different security experts. However, in this report, we follow notions described by Matt Bishop in his book[39]. Computer security generally have three elements: confidentiality, integrity and availability (CIA). Confidentiality is the concealment of information or computer resources from unauthorized access. Information covers all the data stored and managed by the system, and computer resources include processes, data in memory, input-output ports, etc. Integrity is the trustworthiness of these resources. This implies that resources have not been modified by unauthorized access and that they are what they appear to be. Finally, availability is the ability to use the computer resources by authorized access. The quantitative specification of security goals for these three components is called the security policy, which specifies the intended level of security for a system.

A computer security policy defines what actions are allowed and which actions are not allowed in a particular context. The action could be read, write, execute, copy, etc., while context may be the name of subject, object, time, location, trust level, etc. Thus, a security policy defines what is secure and what is not secure, in terms of confidentiality, integrity and availability (CIA). Security policies are enforced by computer security mechanisms, which implement the actual method, tool or procedure. Security mechanisms may prevent, detect or recover from any violation of the security policy.

A threat is a possible violation of the computer security policy by an attacker exploiting a vulnerability in the computer security mechanisms. Threats can possibly cause damages. Threats can be of various types and some that are particularly relevant to nomadic users are listed here. Deception is acceptance of false data by computers causing damage to the integrity of its resources. Snooping refers to interception of unauthorized data resulting violation of confidentiality. Spoofing is impersonation of one entity as a different entity, which in turn allows unauthorized access causing damage to integrity and confidentiality. Spoofing should be distinguished from Delegation, in which one entity authorize another entity to act on its behalf. Denial of service is making a computer resource unavailable to an authorized user for a period of time. Repudiation is denial of responsibility of some action.

As stated earlier, any security mechanism enforces a security policy. This security policy defines the intended level of security for a security mechanism, which we call the 'Designed Security' for that particular security mechanism and is shown in Figure 2. However, the actual security we achieve in practice is less than or at most equal to this intended level, we call this the 'Effective Security' level. Moreover, effective security tends to decrease with time, mainly due to the discovery of more vulnerabilities in security mechanisms, advances in technology, improved techniques in cryptanalysis, etc. According to Gorman[44], a secure system is strong if the cost of attack is greater than the benefit obtained by an attacker. Thus a drop in the effective level of security implies that the cost of attack decreases with time, which causes many more attacks to become feasible. It is important to note, however, that the curve shown in the figure is for illustrative purposes only; the actual shape of the curve depends on the type of the system, the configuration and operation of the system, and parameters of the computational

environment in which the system is deployed.

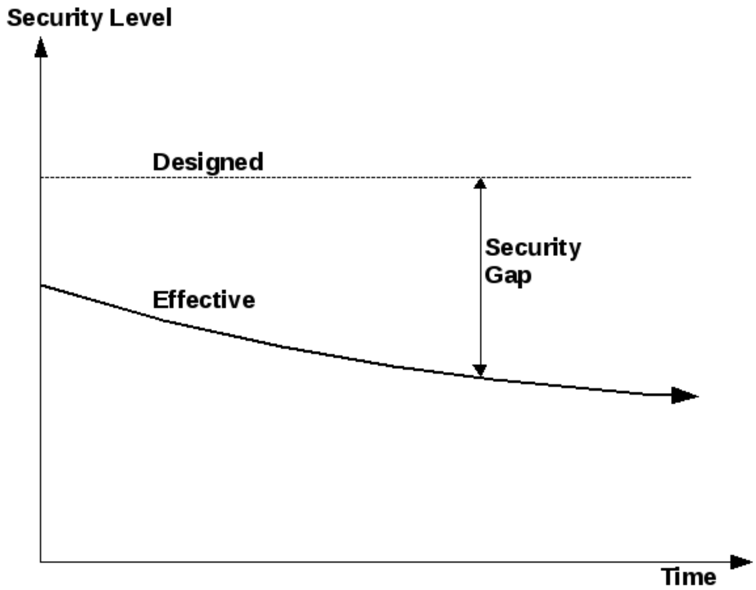

Figure 2: Difference of Effective and Designed Security Levels

There are many different mechanisms designed to achieve computer security, e.g. physical controls, secure operating systems, secure programming, cryptography, information flow control, etc. However, in our research we have considered only the most common and conventional way of computer security, and it is also very well applicable to our focused class of nomadic users. This mechanism is access control provided by UNIX and UNIX-like operating systems. In the next section, we look closely at access control mechanisms and how they are used to achieve security in a nomadic computing perspective.

Access Control

In general, access control mechanisms control whether a subject is allowed to access an object. In the computer security context, subjects are active entities of a system, such as a user, a thread, another computer, etc., that performs operations on objects. While objects are usually passive entities in a system, such as files, memory, computer display, printer, etc., that are operated upon by subjects. For the sake of brevity, we use the term access control, to refer to its computer security perspective and it is shown in Figure 3. A subject requests an access for a object in a system. At this point, the reference monitor get involved. It reads security policy of the system and decides whether this subject is authorized for the requested access. If the subject is authorized, access is granted while in the opposite case access is denied.

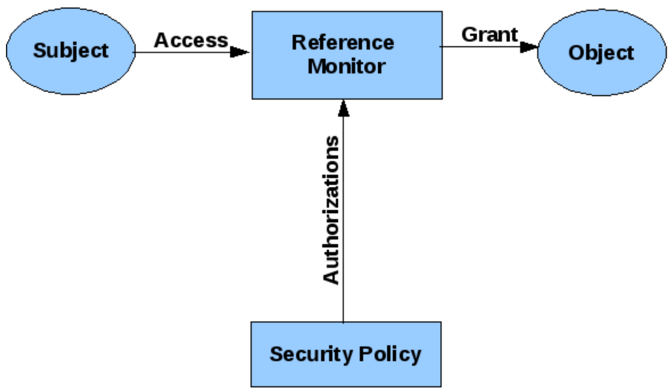

Figure 3: Access Control

	Object-1	Object-2	Object-3
Subject-A	R	R	R
Subject-B	-	RW	-
Subject-C	RWX	-	RWX

Figure 4: Access Control Matrix with Three Subjects and Three Objects

The access control matrix[55] is a formal model for describing a security policy. It lists access rights of each object with respect to each subject in a matrix form. An example access control matrix is shown in Figure 4, where each row corresponds to rights of a particular subject in the system. Each column is called access control list (ACL) of a particular object. As shown in the figure, Subject-A has read access to all three objects, Subject-B has read and write access to Object-2, and Subject-C has read, write and execute access to Object 1 and Object-3. However, access control matrix does not describe fine level details of its implementation or interpretation by reference monitor and thus can be used with a variety of reference monitor architectures.

The classic implementation of the access control model on Unix and Unix-like systems uses access control list (ACL) derived from access control matrix. An ACL is associated with each object in a system and lists subjects who can access this object along with the nature of access, e.g. read, write, copy, etc. The grant of an access request is based on authorizations, which this model gets from the security policy. But, in order to work securely, an ACL based access control mechanism must be supported by another mechanism which verifies the true identity of a subject in order to avoid spoofing. This additional mechanism is known as authentication and authentication must be performed before an access control mechanism is invoked.

In the strict technical sense, the authorization mechanism, only refers to the way in which authorizations are defined in security policy and it does not refer to actual enforcement of this policy. However, in common usage[30][31], the term also includes enforcement of policy and so do we for the purpose of the report.

Another important term related to authentication is delegation, which is a part of the proposal presented in this report. Delegation refers to handing over access rights to another person to perform an activity and is usually for a shorter period of time. Classically, delegation is achieved either by assigning individual permissions of objects or re-associating roles between subjects.

Following is an overview of different types of authentication mechanisms that are used in nomadic environments.

Authentication

All authentication mechanisms are designed to serve one purpose and that is to find out whether the claim made by a subject about his identity is true or not. It may be defined as, "the process of verifying the validity of a claimed user"[44]. User authentication mechanisms are generally divided into two classes.

1. The first class is called human-centric authentication and it involves recognizing some intrinsic or pseudo-intrinsic properties of a human and finding the corresponding identity. Dhamija and Perrig[37] split this class in knowledge based and biometric authentication, but we consider knowledge as pseudo-intrinsic property of a human, which can be acquired and becomes a part of human nature. An example of pseudo-intrinsic property is human memory, where a person remembers a passphrase or knows the solution to a puzzle. On the other hand, human voice is an

intrinsic property. If a voice recognition system is installed at an access point then the system captures the person's voice, extract relevant voice patterns, then it matches these patterns with patterns stored in a local database to find the identity of a user. Few other examples that also utilize intrinsic properties are video tracking, iris recognition and fingerprint matching.

2. The second class is called device-centric authentication and in this case a human is not directly authenticated. The authentication mechanism authenticates a device on behalf of a user with following underlying assumptions;
    1. The user has the authorization to use the device on his behalf for the authentication purpose. This is because a computer has no way to figure out the current association between a device and a user. Thus it usually assumes the previously registered association, which may not be valid as in the case of revocation.
    2. The user is in the possession of device at the time when authentication mechanism is being invoked. It may happen that the association between a device and a user is still valid but for some reason the user is not intended to be authenticated. For example in the case of theft, an unauthorized user may try to get authenticated even when actual user has not intention to do so. In this attack, the authorized user is not possessing the device, although he is associated with it.

The devices used in device-centric approach could be an RFID tag, mobile phone, PDA or even a software program. It is also possible to utilize some hybrid mechanism which may be a combination of human-centric and device-centric behavior. However, individual elements of a hybrid mechanism are recognizable to be classified under one of these classes.

Generally, human-centric authentications require some sort of user involvement and especially in the case of knowledge based authentication, an active user interaction is required. This type of usability constraints limits the maximum frequency of authentication. For example it is quite difficult to authenticate a user after each 500ms using password, voice or fingerprints. However, it is trivial in the most of device centric authentication techniques, for instance a repeated scan of RFID tag after each 500ms is quite common. However, device centric authentication is relatively less trusted approach, as there is always a risk of devices being stolen, cloned or tampered with. Thus, in order to achieve a high level of trust while also allowing frequent authentication, a hybrid approach should be used, which balances combination of device centric and human centric techniques.

During the past four decades, general human computer interaction has evolved from machine centric towards user centric computing. This evolution may be observed from both machine or user perspective. In the following section, we describe the important characteristics at different phases of this evolution in order to emphasize the importance of nomadic users, as nomadic users are the focus of this research.

Evolution of Human-Computer Interaction

As indicated earlier, human computer interaction has shifted towards being more user centric. By looking at it from the angle of computers, we may distinguish three distinct computing paradigms: mainframe, personal and ubiquitous computing. These paradigms represent how computer systems are constructed. From the angle of users, we identify different classes of computer users. These classes correspond to the way in which users satisfy their computing requirements.

Computing Paradigms

By looking at the history of computing in the last four decades, and then analyzing how computing facilities are offered to users, we can distinguish three distinct computing paradigms. These three paradigms, which represent models of interaction between human and computer, are also identified by Allan Kay and Tomei[24], and Weiser[28]. These are mainframe computing, personal computing and

ubiquitous computing paradigm. It is also interesting to highlight that firstly these paradigms co-exist from almost start of the computer age. Secondly, a single user may belong to more than one paradigm at different time instants depending upon his current computing needs and the available infrastructure. However, what has changed over this period of time is relative proportion of users who belong to a specific paradigm. This can be visualized in the Figure 5. This is based on some tendencies indicated in the Cambridge Encyclopedia[50], Snaith [27]and Weiser[28]. However, this graph does not reflect any quantitative data and is drawn here only to indicate a trend, which is mainly caused by advancement in silicon technology.

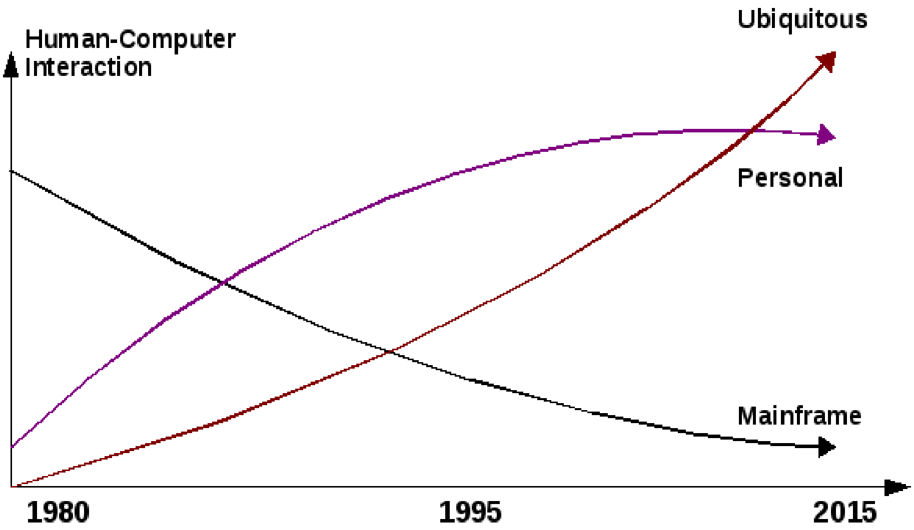

Figure 5: Relative Contribution of Paradigms in Machine Computing

In early days of computing, mainframe was a predominant paradigm and it represents a relation between multiple users and one machine. In mainframe computing, a single powerful machine is being shared by several users on time or resource basis. Even today, according to Ebbers at IBM [25] and Snaith[27], mainframe computes play their role in the world's largest corporations, including many Fortune-1000 companies. Other forms of computing are being used extensively in most businesses, but the mainframe maintains a coveted position in some of e-businesses.

Although personal computers were being built in 70s and early 80s, e.g. Apple II series and IBM PC with 16-bit 80286-processor, etc., but in the late 80s personal computing start becomes the dominant paradigm, marked by the rise of the number of world wide PC users beyond fifty million[40]. The core notion of personal computing paradigm is the concept of one computer for each user. In other words a user owns a computing machine which satisfies his computing requirement[28]. This paradigm includes both fixed and mobile computing machines. Typical examples are classic desktop computers, laptops, PDAs, etc.

More recently the ubiquitous computing paradigm starts taking a big share in computing. Ubiquitous computing (also known as pervasive computing or ambient intelligence) is a paradigm where computing machines are embedded in our environment[29]. Ubiquitous computing represents a human-computer interaction model where devices are expected to be small, inexpensive, robust, networked, embedded and distributed in the environment where computation might be required.

We would like to highlight qualitative nature of properties which define a ubiquitous computing paradigm. Being small, inexpensive, robust and distributed are those features which can not be measured in order to distinguish a clear boundary of this paradigm. In practice, we have numerous environments which are considered ubiquitous in one perspective but not in other. A common example is the computer networks in hospitals. Typically, terminals are networked, distributed, shared and available at all those places where they might be required. But on other hand, in most of the cases these terminal are desktop computers which need an interactive user engagement to invoke the user's session. To avoid this

misinterpretation and to clearly highlight our target group in ubiquitous paradigm, we have provided a more precise definition of an environment used by nomadic users in the next chapter.

Similar to the evolution of computing paradigms, there is another evolution from users' perspective.

Computer users

As computers evolve, so does the way that people satisfy their need for information or computation from computers. In the early days of computing, a user had to physically go to a stationary machine which was installed at some fixed location in an infrastructure in order to satisfy his computing needs, so we call them stationary users. When technology allows manufacturing of light weighted devices along with well connected wireless communication infrastructure, then stationary users evolved into mobile users. These users can carry their computing machine with them and carry out their computing jobs even when they are on the move. More recently, we have started embedding computing devices in the environment, which gives birth to the third evolution phase. We call them nomadic users, characterized by the fact that they have computing machine available to them where ever they need it and thus do not need to carry their computers with them. In fact, the evolution of computing and the evolution of users have re-enforcing effects to each other. It happens most of the time that users tend to use best available computing machine, but at the same time their desire to make use of technology in a particular way forces computing machines to be built on a particular pattern. Thus these two evolutions reciprocate each other to a large extent.

We define a stationary user as a person who uses a computing device which is at a fixed location in an environment, and thus his movement in the environment is confined by the location of the computing devices. In other words when a user needs to compute something, he has to physically go to the location where a computing device exist in order to perform his work. This could be due to the device which is not designed for mobility or due to lack of communication infrastructure which does not allow the device to be moved. Stationary users belong to either mainframe or personal computing paradigm; typical examples are users who use desktop personal computers or bank's ATM machines.

Similarly, as a mobile user fulfills his computing requirement by a device which he carries around, thus these users are characterized by the mobility of computing devices and communication interfaces. Mobile users belong to Personal Computing Paradigm. In this case, a user carries a mobile computing device whose seamless mobility in infrastructure is of concern, in order to provide him with persistent quality of service while on move. Typical example in this class are GSM mobile users, laptop users, PDA users, etc.

Third phase of evolution corresponds to nomadic users, who neither carry a computing device with them nor their mobility is confined by location and availability of stationary computing devices. These users use devices embedded in the environment in order to meet their computing requirements. As their movement should not be confined which implies that environment should have computing devices at all locations where they might need it. These users essentially belong to Ubiquitous Computing Paradigm. However, to avoid confusion with a true ubiquitous environment, we introduce the term 'Nomadic Environment'. It as an environment where computing devices are available at all those places where they are required but their user interface is not ubiquitous. It means that a user has to actively engage with a device for the purpose of computation. Nomadic environments may exist in hospitals, military and many other places where there is an inherent requirement for mobility and the communication infrastructure is more ubiquitous.

Currently not all the computing environments support nomadic users and thus a person may act at different level of evolution, depending on the availability of infrastructure and personal preferences. For instance, a person working in a nomadic environment at a hospital, may also be a mobile user while calling a friend from his cell phone. In the next section, we have presented some properties of nomadic users in order to define our focused class more clearly.

Nomadic Users

There is no canonical definition for a nomadic user and this term also conveys different meaning depending on the context. However, for the purpose of this thesis we define a nomadic user based on certain properties. Although these properties exclude some cases which might be considered nomadic, it is necessary to narrow the scope of our work. We also present some examples to illustrate our definition. In order to distinguish nomadic users from a closely related class, we define ubiquitous users as those computer users who do not interactively involve with computing machines to invoke their unique sessions.

To classify a person as a nomadic user, following properties are compared in a sequence. A given property is either necessary or sufficient. All nomadic users must hold necessary properties which come in the sequence. If a sufficient property is satisfied then we do not compare further properties in the list. However, a violation of a sufficient property can be ignored and the sequence must be continued. These properties are listed in the sequence as under.

1. 'A nomadic user only interact with a nomadic computing infrastructure'. This is a necessary condition and it means that a person can be nomadic at one time instant and may not be nomadic at a different time instant. For example, it is very common that people carry mobile phone with them. If such a person is in a nomadic environment, he is a nomadic user while using devices embedded in the environment, but belongs to the class of mobile users while talking to a friend on a mobile. This is because the communication infrastructure for mobile network is not a nomadic environment. This distinction is also necessary because its rarely happens that a person spends all of his time acting like a nomadic user. So one can say that at a given instant of time a user is either nomadic or not.

2. 'A nomadic user only uses potentially shareable computing devices'. This is also a necessary condition. The device used by a nomadic user is not owned or physically in the possession of a nomadic user and thus the definition only includes those devices present in an infrastructure which can be shared with other nomadic users. All nomadic users hold this property because otherwise the use of device is categorized in personal computing paradigm. For example, if a user is working on a personal laptop, PDA or desktop, we do not consider it as nomadic use case. The security vulnerabilities and usability issues in these types of use cases are different from a sharable device and thus beyond the scope of our research.

3. 'A nomadic user works in his unique session (desktop in classic sense) on a device present in a nomadic environment'. Again, this is a necessary condition as we we can create a nomadic environment where users do not have their unique sessions. If there is no association between sessions and users of a computing environment then the accountability and usability problems are present at a broader infrastructure level. We consider these issues beyond the scope for this research.

4. 'A nomadic user is capable of delegating his duties to another person'. This is sufficient but not a compulsory condition, as working conditions of a nomadic user may not require such a delegation. However, if someone is delegating his own session to another person for a period of time, he is certainly sharing his device and belongs to the class of nomadic users. Usually, nomadic computing environments are highly dynamic, so it is important to support flexible means of collaboration, which implies delegation.

5. 'A nomadic user is conscious of at least one usability case that is the part of a security mechanism'. This is a necessary property and it distinguish a nomadic user from an ideal ubiquitous user. On the contrary, if some authorized user is not conscious about usability at all, there is a very little probability that he would try to bypass the security mechanism. This property can be argued as user is nomadic no matter whether he is conscious or not. But for the purpose of the research we exclude those users who are not concerned to any of the usability constraint and concede them as ideal ubiquitous users.

Let us take a classic example of a doctor working in a hospital, and we also assume that the hospital has enough ubiquitous computing technology to be considered as nomadic environment. This means that it contains various terminals, which are networked, at different locations like wards, conference rooms, office, etc. Let us start applying our defining properties to a doctor in the hospital.

The first property only limits the definition of nomadic users to instantaneous computing actions in a environment. This means we must only look at instants when the doctor is actually using hospital computing infrastructure. Second property is fulfilled as these terminals are shared among other doctors and nurses. Usually, doctors are dealing with sensitive health data related to patients so for accountability purpose each one has his own unique session. This fulfills the third condition for the doctor. The fourth property also holds due to the nature of hospital's job. Doctors often need to delegate their duties to each other, as patient may need a doctor any time around the clock. The last condition is also fulfilled, as for each use of device the doctor has to login and logout. Thus, these repeated interactions are usability constraints which are directly linked to the authentication mechanism.

Nomadic Environment

Similar to nomadic users, the term nomadic environment is also not well specified in the literature and its meaning mostly depends on a particular context. For the purpose of the report, we consider a nomadic environment to be part of ubiquitous paradigm. However, we distinguish it from an ideal ubiquitous environment. We characterize an ideal ubiquitous environment with two properties. Firstly, from the communication perspective, a ubiquitous environment supports a seamless networking of all computing devices. Secondly, from the computing device perspective, it supports a seamless interaction with users that does not involve conscious engagement of any user to invoke a particular session.

We define following two properties to identify a nomadic environment.

1. 'A nomadic environment has communication infrastructure similar to ubiquitous environments'. This implies a communication infrastructure that is well connected and networked. It allows nomadic users to frequently move around within the environment and access their unique sessions from any terminal of the environment.
2. 'Computing devices in nomadic environments are not ubiquitous'. This property distinguishes a nomadic environment from an ubiquitous environment As devices are not ubiquitous, so users interactively authenticate and use their sessions.

Nomadic environments exist at many places, e.g. networks in hospitals, universities, software companies, supermarkets, etc. These environments have well connected communication infrastructure in form of a wireless or wired LAN, but devices used are mostly classic desktop computers. It is interesting to consider that these devices, which are designed for personal computing paradigm, are being used in ubiquitous computing paradigm.

For further elaboration of our definitions, let us take a specific example of hospitals, once again. We have referred to this example at many other places in the thesis for explanation of different concepts. However, by no means, our work is targeted specifically for this particular example environment We aim to make implementation generic and unbiased for the whole class of nomadic users.

Example: Hospital

Usually, modern hospitals have ubiquitous infrastructure in the form of well connected network with centralized access to electronic patient's record (EPR). Most of the computing devices are some kind of computer terminals, distributed around in a hospital environment. Typically these terminals exist at following locations.

1. A doctor's work place has a couple of terminals. They are necessary as doctors may have to examine patients in their office or may have to study or do some research work.

2. Conference and meeting rooms have their own terminals. It is also typical that besides a large display, for presentations and illustrations, there are also terminals in front of each attendee.
3. Mostly, nurses have terminals at their work place to keep track of patients, dosage and other administrative matters.
4. Pharmaceutical or drug stores have few terminals of their own. These terminals are connected to central data base so that they can access the information about prescription of medicines which a doctor recommends.
5. Emergency rooms have associated terminals with each bed which may be used for quick access to patient's history and other medical record.
6. Operation theaters normally have few terminals to assist surgeons. In some cases these are used for remote assistance during complicated surgical operations.
7. In wards, there are terminals behind each patient's bed to show patient's history and current prescriptions. It might be desirable that when a doctor visits a patient in a ward the relevant data is available in a particular context.
8. Some hospitals also provide terminals in ambulances which are connected to the hospital network. For example this may be used for triage or in advance preparation in emergency room for heart patients.
9. In staff cafeteria, there could be a couple of terminals.

It is interesting to look at how nomadic users work in the hectic environment of hospitals. For example, a doctor examines dozens of patients. He responds to emergency calls and carry out surgical operations. After every few days, each doctor treats almost completely different set of patients and possibly at different location in wards. As hospitals operate twenty four hours a day, so typically there are three or four shifts. One can imagine that how computing devices are being shared within single shift and across shifts. Moreover, these nomadic users handle sensitive health data of patients.

In a nomadic environment, where non-ubiquitous computing machines are being used, security mechanisms are also non-ubiquitous. Most prominent are knowledge based authentication mechanisms as they require active user involvement in the process such as commonly used password based login. In the next chapter, we have described that knowledge based authentication mechanisms are not suitable in nomadic use of computing.

CHAPTER - 3

Authentication in Nomadic Environments

If we recall our classification of human-computer authentication from the second chapter, we realize that human centric authentication poses direct usability constraints as it requires the involvement of a user. A mechanism based on human centric authentication can use either human intrinsic property or knowledge. For intrinsic properties like face or voice recognition, it is not necessary for a user to interact consciously although involvement in the process is still required. On the other hand, to make use of knowledge for authentication, a conscious engagement is mandatory, as it is very hard to directly extract knowledge from a mind with out using the human body.

To highlight the usability issues involved in the authentication process, we have selected the class of knowledge based authentication for the research. The primary reason of selecting this class is that typically it represents the worst case, as far as usability is concerned, because of the requirement of conscious active involvement. The other reason of this selection is its most common use, as it comes in form of password based login which is part of many popular operating systems such as Windows, Linux, Unix and Mac OS. Thus knowledge based authentication mechanisms are among the best candidates to discover usability constraints.

Security vulnerabilities faced by nomadic users, for instance in health care, have been indicated by many researchers[34][35][36]. In this chapter we have described usability constraints associated with knowledge based authentication and existing access control models. This has provided us with the basis for our proposal and later on for the security analysis.

Usability Observations

From an authentication point of view, nomadic users are an important class, due to their unique usability constraints. They need to be able to frequently login and logout on any terminal in their environment Moreover, they do not own any terminal which are part of their normal work flow. This means that they should be able to share a terminal with other nomadic users but also be able to invoke and work in their own unique session. As nomadic user do not have fixed pattern of work flow, so sometimes they need to work in place of another nomadic user for a short period of time. This might only be for a limited period of time, e.g. going on vacation, sharing some resources temporarily, going for urgent sick leave, etc. This leads to a situation where a quick and easy delegation is much more important. Also, if nomadic users are working on sensitive data of third party, this whole process must support accountability.

We use the term of knowledge based authentication to include password, passphrase, PIN code, challenge-response or puzzle based authentication mechanisms. But from a usability point of view, they are all similar as they all require user interaction and knowledge. A password is a sufficiently long and complex sequence of keypad characters. The authentication depends on the fact that a password is known to a single person. Usually, a password is associated with a user name. We consider password based authentication mechanism to our example hospital's nomadic environment. The typical behavior of nomadic users[36]is summarized in the following points.

1. Nomadic users tend to use short and easy to type passwords[41][34]. This is mainly caused by their attempt to be able to login quickly.
2. Usually when system puts limit on the length or quality of password, nomadic users tend to write

down their current password. This is their attempt to recover the password in a case they forget it. Sometime, they do not want to memorize a long complex password. Kim-Phuong Vu, an expert of human factors in the area of science and engineering writes in her book[51]; “Many users have half a dozen passwords to remember. That’s why the most common password is ‘password.’ The usual solution is to write it down. But how secure is that? Practicality wins. The probability of remembering six passwords is not that great. Half the people who say they never write down their passwords need to have their passwords reset because of forgetting.”

3. Nomadic users tend to not logout in hope to come back and find their session ready. This is because if they logout on departure, they would need to enter password again which might be considered tedious. On some terminals there is no support for multiple sessions and consequently, on such terminals, login and logout time is considerably high.
4. Nomadic users tend to forget to logout as they have to login and logout many times a day. It is quite natural that a person starts forgetting more and more with age[42].
5. Passwords of nomadic users may be sniffed by mere observations as they frequently need to type them in. It is quite common that when a doctor has to use a terminal, he may be accompanied by number of people, e.g. medical interns, patient's relative, nurses, etc.
6. Nomadic users tend to give away their password, PIN codes and even authentication devices to their colleagues for delegation purposes. This is due to the fact that nomadic environments are usually dynamic and frequent collaboration is necessary. The easiest way is to to give away one's authentication token which is normally practiced by nomadic users.
7. Sometime, nomadic users share authentication token in a group. This might be useful to work efficiently in a group.
8. Due to time constraints, it sometimes happens that nomadic users are unable to properly delegate their position using individual permissions or roles. If we imagine a doctor in a hectic environment, it may become very difficult to figure out what permissions or roles are necessary to perform a certain job. Easiest way could be to give authorization of every thing, but this may not be possible in some implementation
9. If authentication token was shared previously and some thing goes wrong, then no one is accountable. This might be a honest situation where a person, who has performed a certain action, simply forgets.
10. If passwords are shared, revocation becomes difficult. One option is to change the password but this raises the problem of simultaneously propagating the new password to all authorized users. The frequent change of password might not go indefinitely and users may start repeating previously used passwords, besides they need to remember the most recent password.

Now let us analyze possible security vulnerabilities which arise as a result of applying classic authentication mechanisms to nomadic environments. Before we start, let us consider some example specific factors. Terminals available in hospital ubiquitous environment are designed to be sufficiently secure with the help of password based authentication. In certain nomadic environments, like hospitals, nomadic users are dealing with sensitive data related to health records of individuals. This means every user should have his unique session and that all actions taken by individuals should be recorded, so that the individual may be held accountable at later stage if necessary. It is important to note that there is no fixed pattern of work due to variation in number of factors, for example number of patients at a given time, nature of their disease or sickness which may influence their assignment to doctors and nurses, arrival of emergency, urgent operation, etc. These factors dictate that the system should not be constrained by the usability.

Following are some of possible vulnerabilities.

1. Even in normal working conditions, passwords are always subject to numerous attacks and this is

more true for nomadic use[44][46][47]. If a password is short and easy, it might be possible to hack into system using some dictionary based brute force attack or commonly available password crackers such as 'John the Ripper'[8]. This poses a considerable threat in an environment where sensitive data is present. For example, if some one hacks into hospital data base, he can view and manipulate patient's data in the electronic patients record (EPR), possibly to blackmail the person at later stage. If it is emergency rescue environment, one can trigger false alarms that may further be used by other purposes e.g. terrorist activities.

2. The common problem with classic knowledge based authentication is that people tend to forget authentication secrets that they are supposed to know[37][41]. This may cause denial of service in nomadic environment. For example, if the doctor on duty forgets his new password, he would not be able to access patient's medical history or prescribe some medicine. This could be a very serious situation even if it persist for a short period of time.

3. Writing a long password somewhere is the violation of very basis of human centric authentication mechanism and is quite vulnerable to stealing. Again, attacker can use a stolen password for hacking or may use it for guessing the password of same person for much more sensitive data like his bank account. For example, if an attacker knows the password of a user's email, personal computer or mobile device then it might be much easier to guess the password of his laptop. This is because people tend to use same type of passwords or repeat particular patterns.

4. Usually, logout is not considered as part of authentication but, similar to Bardram's proximity based login[11] and Corner and Noble's zero interaction authentication[7], we consider it as integral part of the overall authentication process. If the user does not log out when he goes away then, from security point of view, it is a failure of authentication mechanism because it allows any one to be authenticated as the user even when the person is no longer there. This situation is vulnerable for various attack, for example attacker may install a Trojan horse on a pre-authenticated system or may change the password causing a denial of service attack.

5. Password sniffing is a common problem and people, who are not good at typing, are especially vulnerable to this. This vulnerability is hard to avoid and in the long run, it is quite independent of using lengthy or complicated passwords. Once a password is sniffed, system is open to many local and remote access attacks.

6. Giving away one's password is like making a copy of the authentication token and moreover, allowing the other person to make further copies of that. This could be considered as the worst form of delegation. If the person who gets the password is not careful, this might be stolen by an attacker who with stolen information may damage the reputation of the owner of the password. Also, from group dynamic perspective, the owner of the password may start losing trust in the other person which may effect work efficiency.

7. Sharing password in a group or using a group password is also vulnerable to various attacks. One problem with this approach is that no one is accountable, although every one is responsible and authorized. This is a typical security vulnerability where people have authorizations but no responsibility and attacker may use this fact to create chaos. Let us take an example of a password which is shared among a group of doctors who work at different time in a week. If an attacker knows this and has the capability to tap landlines, it might be possible to crack the system. For instance, the attacker may send a fake message to all group members that someone outside the group knows the password. With non-zero probability, this may cause the group to change the password. Most probably, the doctor who is currently on duty would change the password and would try to inform other group member about new password. In this case, attacker can sniff new password while it is being distributed to other group members on a land line.

8. The essential part of delegation process is revocation. Otherwise, one can imagine that there would be continuous spread of authorization in an organization and a time may come when every one is authorized to do anything. Revocation is difficult to achieve in password based authentication. A failure in revocation may cause the possession of permissions which one is not

authorized to have and this may go unnoticed for quite a while. An attacker may use this fact for unauthorized access, causing damage to the system. Accenture Terminated Employee Survey[41] shows that about 10% of ex-employee access the databases of their former employer. This is because employers were not able to properly revoke their permissions.

9. In classic non-persistent methods, if delegation is revoked, it usually takes effect when a user tries to re-authenticate himself. An attacker may steal the session and continue to use it indefinitely with out logging out.

10. Active user's involvement in an authentication process consumes lot of time when we add small chunks of time, consumed during login and logout, for all users. This is especially true if terminals do not support simultaneous multiple sessions. Apparently, this is a matter of efficiency in organization but attacker can use this fact to create a vulnerability. For example, one can create a situation where users need to login and logout at much higher rate, thus reducing their productivity and causing a partial denial of service attack.

After analyzing these use cases and security vulnerabilities, we can infer that the systems which are designed to be secure, become much less secure due to usability constraints. Due to these usability issues, people tend to avoid security checks. Thus for nomadic environments, it does not make sense to build highly secure systems that are difficult to use, as it finally leads to defeating the initial goal of security. An attack on system security is always expected from the weakest point and this does not necessarily have to be the most obvious or the most well protected point by a security mechanism[45]. Thus effectively, above listed vulnerabilities are points of attack and must be addressed.

Previously, we have analyzed classic authentication from usability perspective. Now, let us start analyzing existing access control models in general, to find out possible vulnerabilities for nomadic use.

Existing access control mechanisms

Before looking at existing mechanisms, let us divide the range of security vulnerabilities, resulting from usability, in two parts. First part is related to how a nomadic user is authenticated in nomadic environments. Second part is related to authorization management, which comes after a user has been successfully authenticated. However, in the second part our focus is only on the delegation issues. This is because, firstly, other authorization management functions do not play sufficient role to affect the usability of a nomadic user. And secondly, the focus of our thesis research is not to find all factors, but only the major usability factors that cause security vulnerabilities in a system that uses conventional access control mechanisms. In our case, these factors turns out to be authentication and delegation.

In the following text, we start analyzing nomadic users in the context of Role Based Access Control (RBAC). This is followed by the analysis of more classic ways of access control, Mandatory Access Control (MAC) and Discretionary Access Control (DAC).

Role Based Access Control

Security management of large complex networks is quite a challenging task[8] and to address this problem, Role based access control (RBAC) model, as formalized by Ferraiolo and Kuhn[9], has become the predominant model for implementation of access control system. Major IT vendors began developing products based on RBAC as early as 1994[8]. In year 2000, National Institute of Standards and Technology (NIST) adopted it as a standard[10].

Framework

RBAC is an efficient and standard access control framework, and its delegation mechanism is very flexible and useful for information sharing in nomadic environments[1]. In the context of RBAC, roles are identified as different job function in an organization. Individual users are then assigned to these roles

based on their job responsibilities. In RBAC, permissions are always associated with the roles. All users inherit permissions through the roles allocated for them as shown in Figure 6. This role-driven model aims to simplify the management of permissions[2]. Although the group mechanism, which is a part of many Unix-like systems, can be considered as a subset of RBAC framework, the RBAC model does not recognize it. This is because the notion of a session is a critical part of RBAC, but it is not considered in the groups. A session allows activation of some of roles and without this explicit distinguishing, all roles of a user are active which clearly violate the least privilege rule[4].

RBAC model does not specify how a person should be authenticated. As far as nomadic users are concerned, the choice of a specific level in the framework for authorization does not lie in the scope of usability. Which means that we are free to choose any well implemented RBAC model for authorization purpose. But we have to look at the delegation issue in more detail.

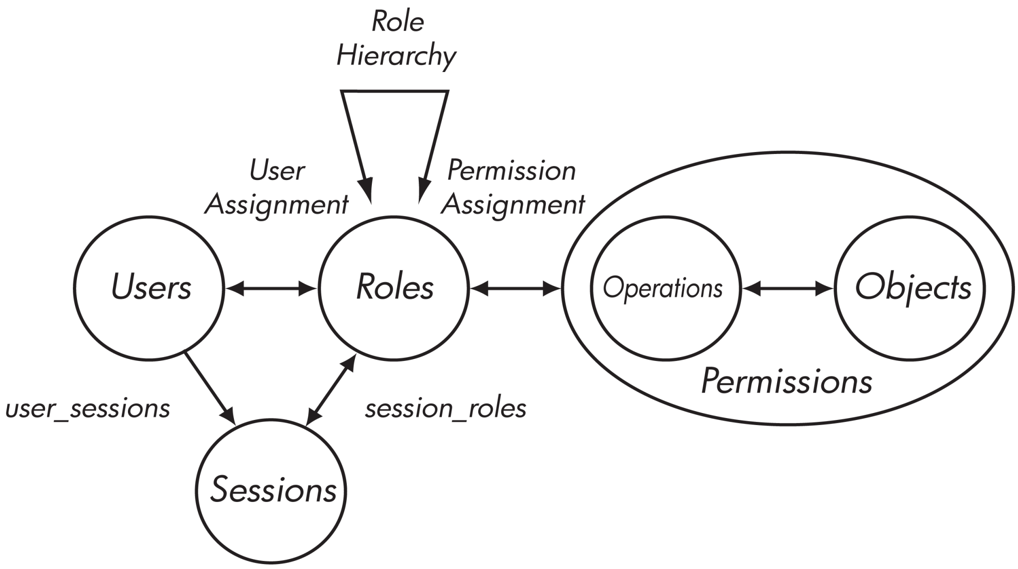

Figure 6: Role Based Access Control, [Courtesy Hewlett Packard]

Delegation

In any system which uses access control, delegation is required when a user needs to act on another user's behalf. A user who delegates his authorization is called a delegator. In order to delegate an authorization, delegator must possess that authorization right, in addition to the authority to delegate it further. Similarly a delegatee is a user who receives authorizations as a result of a delegation operation. With delegation mechanism in place, the delegatee has the privileges to perform intended task without explicit consent of the delegating user[1]. Of course, this implies that the delegator should know what privileges are necessary to perform a task.

By definition, user delegation is a way of assigning authorization available to one user to another user and can be of two type: grant operation and transfer operation[2]. As the names suggest, grant operations share authorization and transfer operations actually transfer them from one role to another. Similarly, delegation may be invoked from two perspectives: administrative delegation and user delegation[2]. Former allows an administrative user to assign access rights and does not necessarily require the ability to use the access right. On the other hand user delegation allows a user to delegate his available rights but the delegator must posses the ability to use the access right himself. Moreover, rights can be delegated at two different levels in RBAC: role delegation or permission delegation[2]. However, it must be noted that individually delegating all permissions and delegating a complete role are different, as in former case the delegatee is not considered to possess the role although he has all the permissions.

From usability point of view there is a problem in this simplified view of delegation in RBAC because it is very difficult to find out exactly which permission are required to perform a certain job. To address

this problem developers can package permissions into a collection, which is called Ability. This Ability must be assigned as a unit to a particular role and hence to user[1]. So in RBAC context, one can define Ability as a collection of individual permissions that must be assigned as a one-unit to a particular role in delegation process.

The exact effect of delegation mechanism under RBAC for nomadic user is implementation dependent. However, there is a common usability problem associated with all RBAC implementations.

- Since the RBAC delegation model[26] manages authorizations in terms of roles and permissions, thus the delegation control is also restricted to roles and permissions. But if we consider the essence of delegation in nomadic context, for instance in hospitals, then it turns out that delegation can also be a user level problem and RBAC model does not support it.

Once a delegation is completed then, at some point in future, this delegation needs to be revoked. Revocation could be non-trivial in some cases. Wang and Li[3] categorize revocation schemes with four dimensions: dependency, resilience, propagation and dominance. Consequently these four dimensions leads to sixteen different revocation schemes in simplified RBAC model. Again, not using delegation at user level makes this process a lot more messy causing usability constraints. Moreover, RBAC also does not guide about persistence of delegation process which leads to implementation dependent vulnerabilities. For example, it may happen that a delegatee continues to use a delegated session using old authorization data.

Mandatory Access Control (MAC)

Mandatory Access Control (MAC), as loosely define by NIST[23], depends upon central security policy written by security policy administrator. A user does not have the right to override this policy in any case. This means whenever a user or some process wants to access some resource like sockets, memory, file system or display etc, system refers back to the centralized security policy to find out whether this operation is allowed or not. MAC is supported in MLS (Multi Level Secure), SELinux (Security Enhanced) and many other military-oriented systems.

Due to nature of MAC, it does not support independent user level delegation and thus it is not very suitable for Nomadic users. This is because, a user in MAC has to refer system security administrator for any change of permissions which is quite undesirable in terms of usability for nomadic working environments.

Discretionary Access Control (DAC)

Discretionary Access Control (DAC), as defined by NIST[23], is based upon the identity of a user or the group to which the user belongs. Additionally, a user who have certain permission is capable of delegating those permissions to another user and do not need administrative privileges. Example of DAC is a traditional UNIX system and UBUNTU desktop distribution of Linux etc. In the UBUNTU every object in system has an owner and owner controls the permission for that object and can change those permissions. MAC and DAC can co-exist on a system and in that case MAC has always the highest priority.

As we may conclude that DAC has the support of delegation, which means users can freely delegate their rights to other users. But there are couple of problems with its implementation in popular operating systems.

1. For delegation purpose, individual resources are delegated and there is no easy way to aggregate resources and thereby delegate all the permissions necessary to perform a specific task in a single operation. Since nomadic users need to frequently delegate thus it is usually difficult for them to figure out relevant resources for a certain task in a short period of time.

2. Delegating individual permission requires a relatively long user interaction depending on the number of resources associated with a job. Due to the requirement of frequent delegation for

nomadic users, prolong interactions with a delegation mechanism motivate them to circumvent the default delegation mechanism besides reducing their work efficiency.

3. Usually, there is no accountability in this approach as the system does not always distinguish between authorizations that are granted by the system administrator and the authorizations that are obtained by a delegation process.

4. This delegation is at resource or group level which means that delegatee still works as delegatee even after delegation. If a user A delegates a resource R to a user B, to perform certain action on R on behalf of A, then still the delegatee B performs those actions on R as user B, although B is now capable of doing those actions. This might not be desirable in some cases.

5. There is no guidance about how a user should be authenticated, causing implementation dependent vulnerabilities, for instance various usability problems with password based authentication of nomadic users[36] as described earlier.

After reading these usability problems related to authentication and delegation, one may feel that there should be additional usability requirements which must be considered while designing a security mechanism for nomadic use. In the next chapter, we have listed some of these usability requirements and later on transformed them into an authentication framework.

CHAPTER - 4

Authentication Framework

It is clear from the discussion of previous chapters that the classic methods of knowledge based authentication are not suitable for nomadic users. This is not because of some security weakness in these mechanisms, but the real problem lies in the situation where they are being used, and we refer it as the usability constraints of authentication mechanisms. In addition to authentication, many other security mechanisms might also be unveiled by a more in depth and thorough analysis of nomadic users. But, our focus of this research is only on the usability constraints in authentication mechanisms.

To design a security mechanism for a computer system, we usually start with specifying those requirements which this mechanism must accomplish. If this mechanism is meant to secure the interaction between human and computer, typically following questions are answered, in order to enumerate relevant security requirements.

1. How to verify the identity of a user for a particular session?
2. What is the temporal pattern of invoking a session?

Conventionally, system security designer focus on the first question in order to list these requirements for designing an authentication mechanism because apparently, answer to the second question is related to usability issues only. However, with changing trends of human-computer interaction, this conventional approach is not sufficient. For instance, in conventional password based authentication, security is based on the user's selection of a random password. Authentication is successful if the user enters the correct password and it is fundamentally secure if we assume a sufficiently long, random and secret password known only to its owner. However, when we start looking at temporal patterns of this interaction for nomadic users, we may start wondering about how long and random password a user can have considering that he has to login again and again after every couple of minutes. The conclusion is obvious, a user might be tempted to start using very short and simple passwords. Thus a system, which is normally considered secure, becomes much less secure due to a unique temporal pattern of interaction. Moreover, a nomadic user spends a considerable amount of time on non-productive work (i.e. authentication).

Thus, usability should be considered as an integral part of any authentication mechanism designed for nomadic use, otherwise numerous security vulnerabilities may occur, causing effective security to drop considerably. As authentication is prerequisite to commonly used access control mechanisms, thus constraints present in authentication are implicit cause of many security vulnerabilities found in computer systems. This implies that the effective security level of a system, which uses classic authentication and delegation techniques, has dropped considerably in nomadic environment, due to the presence of those vulnerabilities which we have pointed out in the previous chapter. In the next section, we have listed some usability requirements which we conclude after studying the interaction of nomadic users with classic authentication mechanisms.

Usability Requirements and the Hypothesis

After the study of nomadic environments we may infer that any user interface, which is connected to a security feature, must be considered as an integral part of that security mechanism. We can design a mechanism to achieve the highest level of security, but if the mechanism is hard to use then it may fail to provide the desired level of security in practice. These conclusions highlight the need to take into account the potential user and corresponding environment, before start designing a security mechanism. This approach makes sure that what we design, is actually used.

For usability constrained nomadic users, usability should not be traded off for the gain of security. We

have presented this inference, in authentication context, in form of a list of requirements. These requirements specify the nature of user interaction towards authentication mechanism, in order to avoid security vulnerabilities and thus may increase the level of effective security.

Our hypothesis can be stated as;

"An authentication mechanism designed for nomadic users would be more secure and usable, as compared to knowledge based authentication, if it meets the following requirements."

1. Interaction between a user and authentication mechanism should be minimum. This requirement is introduced to remove the motivation of busy nomadic users to circumvent the authentication mechanism. Although, we can make it absolutely zero by employing some device centric authentication, but there is a security risks involved. For instance, if a user is in front of a computer but does not want to use it then there is no way by which the authentication mechanism of the computer can figure out his intent. Thus some kind of gesture is necessary.

2. Authentication for nomadic user should be persistent. This is due to the fact that we can not rely on busy nomadic users for logout. This requirement is aimed to avoid another usability constraint by shifting responsibility of logout from users to authentication mechanisms. However, the persistence of authentication can be defined in two ways. Firstly, in the case of a computer, it keeps a person authenticated as long as his presence is verified. For instance a system may re-authenticate a user after each second to keep his authentication active only for the duration in which he is present on the system. Secondly, in the case of infrastructure, it keeps a person authenticated by keeping track of the previous authentication event. This tracking may be in form of video tracking or system may issue authentication credentials which the user may use at different access points. For nomadic users, we recommend the former form of persistence, which take cares of logout.

3. Multi factor authentication should be used for nomadic users. As indicated earlier, device centric authentication is very suitable for achieving minimum interaction and persistence But there is always a risk of device being stolen, cloned or tampered with. Thus we can not totally rely on device centric approach in secure environments. On the other hand human centric authentication is more trusted but it is not quite suitable for attaining minimum interaction and persistence. This leads us to the conclusion that hybrid and multi factor approach should be employed in the authentication mechanism designed for nomadic environment.

4. Context information should be used in an authentication process. Although nomadic users access different computing devices in the environment, they mostly do some particular jobs at particular terminals. For instance, in the case of hospitals, a doctor, while visiting a particular patient in a ward, is interested in the patient history, prescriptions and relevant medical record. When same doctor is in a operation theater, the context is different. Thus context information is important and should be used to invoke particular role in order to achieve better usability experience.

5. Delegation should be at authentication level of users and must be persistent with minimal user interaction. This requirement reflects the delegation as it is normally practiced, for instance by giving away one's password or using a group password[34].

First two requirements have already been used in authentication mechanisms in the state of the art[7][11][12][34], but not as underlying usability requirements for nomadic environments. We represent these in form of general requirements for nomadic use. In addition to this, our idea of user level delegation is presented in the form of fifth requirement.

The idea of user level delegation represents exactly how we use delegation in our daily life. If some one wants to give his car to a colleague to visit a market, he would handover the original keys of the car. Similarly if someone wants to lend his mobile phone to a friend for a call, he would hand over whole mobile phone and probably would not start securing his personal data from mobile memory. This is how things work when we are in hurry and want to delegate for a shorter period of time. This is exactly what happens in nomadic environment[34]. Nomadic users are typically in hurry and they tend to give away their

sessions or even passwords if they need a short break. This works because there is a kind of trust relationship between co-workers and colleagues. However, this notion of trust is difficult to represent in the access control mechanism of a computer. So we propose to let them delegate at user level, but log the delegation event for accountability purpose. This does not restrict the use of other delegation techniques and it only impose the requirement of inclusion of user level delegation. The minimal user interaction in the delegation is somewhat vague criteria, but it enables us to compare alternate approaches available for the user level delegation.

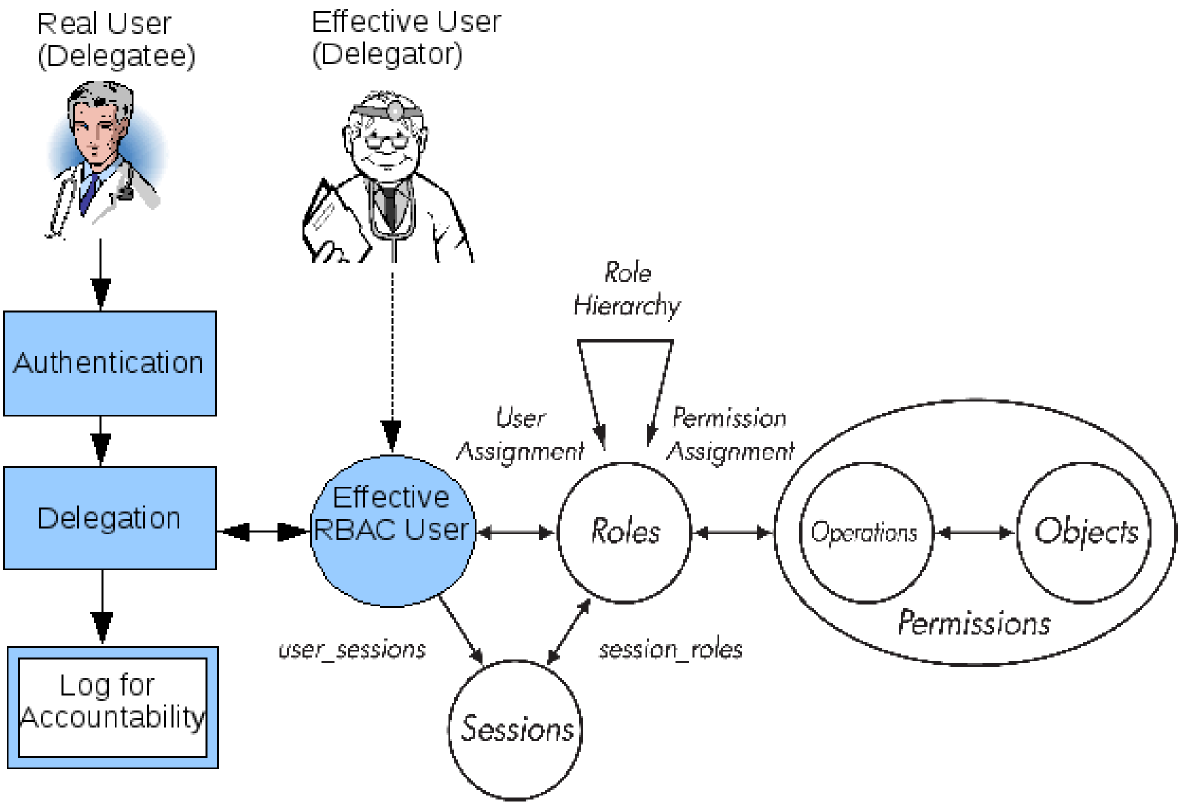

Figure 7: Delegation at User Level [Adapted from Figure 6]

Figure 7 shows how a user level delegation can be implemented if underlying access control mechanism is RBAC. We let RBAC to delegate at permission and role level. However, we provide the provision to delegate at user level as shown in the figure. This is essentially similar to the fact that a delegator gives his password to a delegatee, but in our solution this is secure, accountable and easily revocable.

These requirements are implementation independent and can be applied to any computing environment designed for nomadic users. We can use them in the design of authentication mechanisms for various nomadic environments, e.g. hospitals, supermarkets, emergency rescue services, etc. Also, these requirements are quite independent from the underlying cryptographic primitives or protocols and thus can be used for designing any desired level of system security. We claim that if an authentication mechanism designed for nomadic environment follows these requirements, the effective security of the system is expected to increase, which is illustrated in Figure 8. One should notice that our proposal is not to increase the level of designed security. We are only aiming to remove usability constraints which are causing the level of effective security to drop significantly in nomadic environments. The expected result of our listed requirements is represented as a right bar in the figure, which shows the increase of effective level of security as compared to classic implementation. We have presented justifications for this claimed increase in the Evaluation chapter. However, quantitative value of this increase depends on many other factors including type of system, type of users, frequency of interaction, security policy, underlying cryptographic primitives, etc.

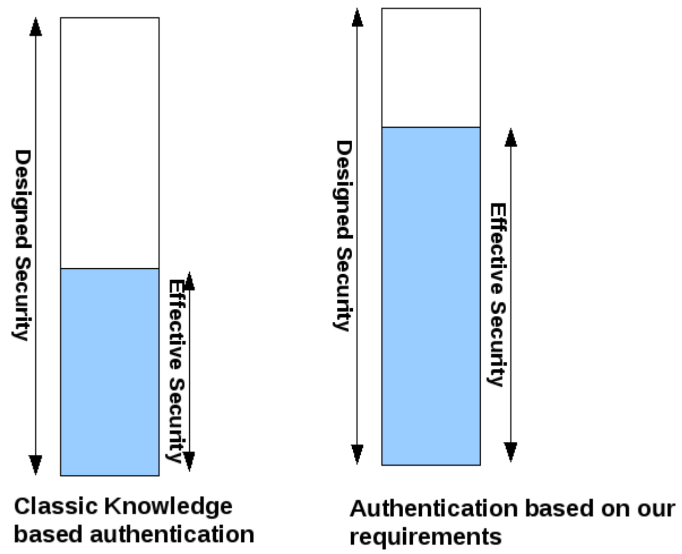

Figure 8: Visualization of Expected Security Improvement

To meet usability requirements of authentication in network environments, we have proposed a reference framework and called it NDAF (Nomadic Delegation and Authentication Framework).

Nomadic Delegation and Authentication Framework

To meet above listed usability requirements to a typical nomadic environment, we have listed following specifications for our proposed framework, called Nomadic Delegation and Authentication Framework (NDAF). These specifications target a wide range of nomadic environments to a large extent.

1. Its design should be based on usability requirements listed in previous section. These requirements are identified in our hypothesis, which we are interested to evaluate later on, using security and usability analysis of its prototype.
2. The architecture of NDAF should be network based. This means it should use client-server approach so that it can be easily ported to existing networks of nomadic environments and support of session migration becomes trivial.
3. It should support multiple active session on any given terminal. This requirement arises as a natural consequence of typical nomadic use, where multiple user share a single terminal. In order to make this use efficient from usability point of view, support of multiple session is necessary.
4. It should be compatible to classic access control mechanisms for computer security. Again, this is to make our reference implementation widely applicable to real nomadic environments. Currently we have selected Debian Linux for this purpose. The world's most popular distribution of Linux (UBUNTU) is in fact based on Debian Linux[43].
5. The framework should not restrict the use of particular cryptographic primitive or protocol used in a authentication technique. In addition to make it widely applicable, this specification keeps the focus of framework on usability, instead of designed level of security.

In order to achieve minimum interaction and persistence of authentication, the most natural and efficient way of authentication for nomadic user is proximity-based login technique, which allows users to be authenticated on a device simply by approaching it physically[11]. However, specific techniques for proximity based login and underlying cryptographic functions are not in the scope of this thesis work. Never the less, the choice of specific techniques is directly proportional to the level of security which we want to achieve but for the research, we are more interested in relative measure of security. It means that

whatever security level we choose, the over all design should not reduce it as a consequence of usability issues of nomadic users. The idea of proximity based login has already been used in many products in ubiquitous computing research with different level of security like Active Badges[13][21], AT&T's ActiveBat[15] which also uses session migration, Microsoft EasyLiving[14] , IBM's BlueBoard[16], AwareHome[20] and Personal Interaction Points[22] which uses RFID tokens for authentication, XyLoc System[17], etc. To support user level delegation, we have designed a custom mechanism, as currently this type of delegation is not part of existing access control models such as RBAC, MAC and DAC.

The basic architecture of our authentication mechanism is shown in Figure 9. The thick line in the middle represents a communication network, such as Wireless LAN, in a nomadic environment. On the left side, some client terminals are shown which are connected to their local authentication modules, like RFID based tag reader, password based login program, bluetooth authentication module, etc. All these terminals run 'NDAF-Client' application which interact with local authentication modules in their native format. For example if this mechanism is password based authentication, then it is responsible for prompting for a login name and a password. If it is a RFID tag, then NDAF-Client communicates with the RFID reader with supported protocol through serial or network ports. All authentication data is sent to an application called 'NDAF Fusion' which runs on main session server.

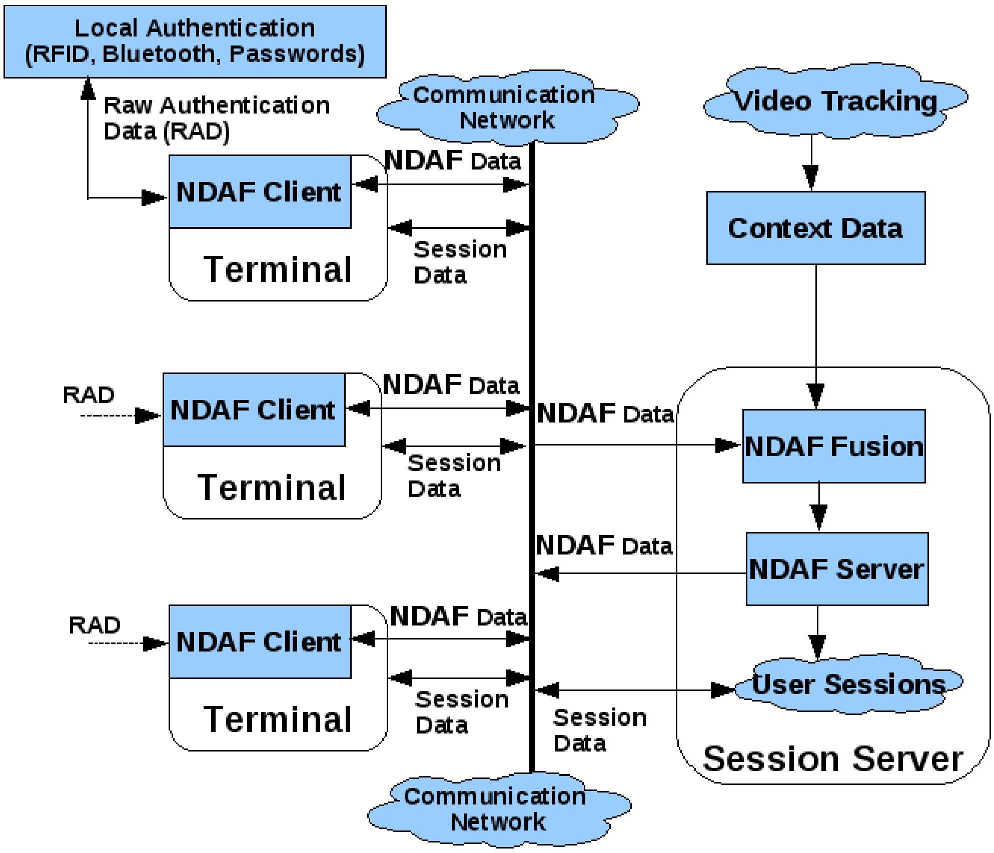

Figure 9: Top-level Architecture of NDAF

The right side of the figure shows the session server. This server can be a single computer or a distributed infrastructure. It runs two applications called 'NDAF-Fusion' and 'NDAF-Server'. 'NDAF-Fusion' is a fusion engine (e.g. Jury framework[48]), which combines all authentication and context data and output authentication status to 'NDAF Server' application. This module represents back end of our multi factor authentication. We may configure this module to implement a specific authentication policy. For example, we can specify that authentication should be granted only if user's badge RFID and his mobile RFID are validated. 'NDAF-Server' is responsible for starting, stopping, locking, unlocking of a user session which can be remotely accessed from a terminal. The decision of NDAF-Server depends on authentication data being received from 'NDAF-Fusion' module. Underlying detail of 'NDAF-Fusion' can

be found in Jonnson's co-authentication framework[48]. Similarly the detail about context data (i.e. user's location) and corresponding authentication techniques can be found in 'PAISE' scheme and related experiment by Kirschmeyer, Hansens and Jensen[53].

It was indicated earlier that most of human-centric authentication techniques are not well suited for persistence and seamless active authentication. We can use device centric approach, but again there is always a risk of device being stolen or tampered with. This is because in device centric authentication a user is not directly authenticated, instead device is used on user's behalf. The best solution could be to combine all strengths of each technique and masks possible weaknesses in usability and security. This leads to a solution where multiple authentication techniques are combined to form a unified authentication mechanism, and may also be called multi factored authentication. If the 'Jury' framework[48]is used for achieving multi factor authentication for our framework, it would look like Figure 10. This framework enables an easy integration of arbitrary many authentication techniques, including biometrics, knowledge based and device centric techniques. Moreover, it is easy to integrate it with wide range access control systems which work on either probabilistic or binary authentication results.

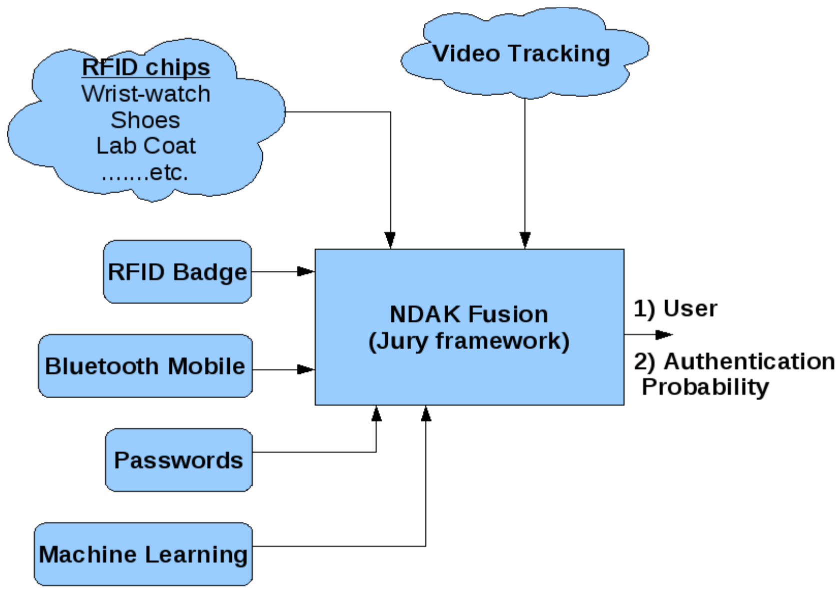

Figure 10: Multi Factor Authentication for Nomadic User

In the figure, we have combined RFID, bluetooth and password based authentication along with machine learning algorithms. A typical policy for this framework could be following.

1. When first time a user enters in the nomadic environment, he should enter his password.
2. The system automatically associates RFID tags present in his clothes, watch, shoes and batch with user identity. Similarly an active bluetooth from user mobile is additional binding parameter. We can specify that whenever RFID batch along with one of additional binding parameter is present, system should assume it as a valid user.
3. Machine learning can be used to avoid login on a system, when a user just passes nearby. Also a computer should not automatically try to login on a system that a user normally does not use.

We have selected client-server approach in designing NDAF and one of the reason for this selection is to support session migration. The term, session migration have different meanings depending upon the context. For the purpose of this framework we define session migration as following, which is also similar to many existing solutions[13][14] [15][16][17][21][22].

1. The system automatically authenticates a user on a network computer at which one is approaching.
2. After successful authentication, an automatic start of relevant session takes place.

3. When the user departs from the scene he is logged off automatically and may be able to start his session at a new place in similar way.

Session migration enables activation of a user's unique session on a computer at which he intends to work. In some sense, his session is following him by migrating from terminal to terminal, depending upon physical location of the user.

Prototype Implementation

To demonstrate the proposed framework, we have instantiated it in the form of a prototype authentication mechanism as shown in Figure 11. The prototype consists of a Intel based desktop computer, running UBUNTU Linux (which is the most popular distribution of Debian Linux[43]) with GNOME desktop. For authentication purpose, we have augmented classic login based mechanism with RFID based authentication mechanism, which represents a branch in multi-factor authentication. The client and server parts of the framework are present on same computer.

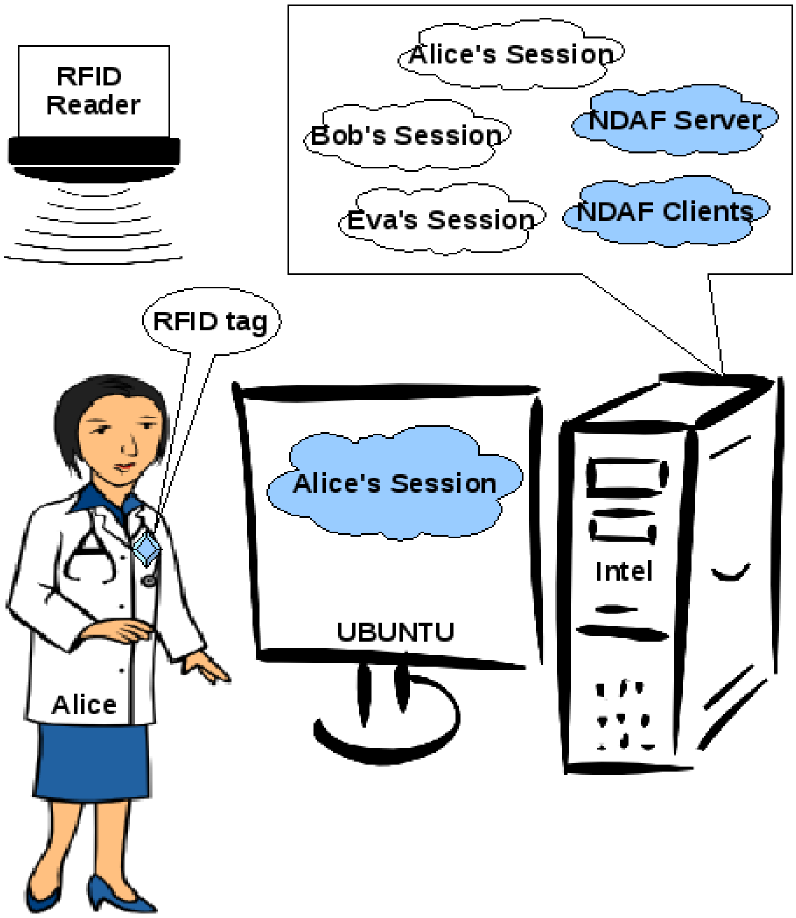

Figure 11: Prototype Implementation[Clip arts: www.clker.com]

As shown in the figure, there are three sessions of users in the computer's memory, which belong to Alice, Bob and Eva. Normally when no one is present, these sessions are suspended and the computer display is locked. When a previously registered user approaches, the relevant session is invoked instantly. A registered user is a user whose radio identification is stored in NDAF Server's database. In the figure the user Alice walks up to the prototype. When she is within the range of RFID reader, her RFID tag is scanned. After the verification of her identity, NDAF Server unlocks and activates her session on the computer's display. Since this process is completed in the fraction of a second, she does not feel that authentication has taken place. Moreover, the authentication mechanism of the prototype provides persistent authentication which implies that she remains authenticated as long as her presence is detected

by the RFID reader. If she departs from the scene authentication is automatically revoked in the system. We have implemented our prototype on a single machine running multiple session, but it is trivial to expand it on a network.

To achieve minimum interaction and persistent authentication, the choice we have made in this regard is device centric authentication using passive RFID tokens. Persistence is achieved by periodic rescan of the tags and updating the authentication state, which is trivial in a device centric approach. One may argue that device centric authentication, especially with passive RFID tags, is not best choice from security point of view, but it is beyond the scope of the research as our aim is not to increase designed security level. Thus particular technology is only a matter of choice for us and we are interested only in increasing effective security by removing usability constraints. Moreover, due to the presence of Fusion engine we are in fact depending on multiple device centric techniques, which give us a much higher level of trust.

We have also implemented user delegation in this prototype according to our initial requirements. It is essentially at user authentication level. The user interface is minimal as the delegator has to issue just one simple command from the terminal to grant or revoke a delegation to some user. Similarly, delegatee can view all available delegations and switch to one of those using a simple command. The underlying mechanism of our delegation approach is distinct from other access control models, for instance RBAC model where delegation is supported at authorization level. Thus, it truly reflects our hypothesis in the implementation. The design detail of the prototype is presented in the next chapter.

Hypothesis testing

The primary purpose of NDAF and its instantiation in the form of a prototype is to substantiate our hypothesis. For this purpose we have constructed some tests to be executed. The evaluation and analysis of results of these tests is presented in the sixth chapter.

1. Active user interaction
    1. One should make multiple users' accounts on the machine. These users should be switched randomly after each minute and this process should continue for a couple of hours. One should calculate the time used in the authentication process.
    2. Those users who are not registered with the mechanism should be able to use their session using default authentication mechanism. This test makes sure that if a user loses his tag then he should be able to use the system, for instance by entering the password.
2. Persistence of the authentication
    1. One should calculate the logout response time when a user departs from the scene.
    2. One should calculate the response time when one user is replaced by another user.
    3. A single active session, with multiple registered user present in the radio field, should be used to demonstrate the mechanism's ability to pick the right user who is currently running the session.
3. Delegation
    1. One should calculate the average time to invoke a delegation and a revocation.
    2. One should calculate the response time for a delegation and a revocation.
    3. A user should delegate his session to another user and then both, delegator and delegatee, should activate the session.

In the next chapter, we have explained the architectural detail of the prototype. The summary of radio frequency identification (RFID), relevant to the prototype, can be found in Appendix A.

CHAPTER - 5

Implementation Details

The Nomadic Delegation and Authentication Framework (NDAF), which addresses the usability requirements of nomadic users, serves as a reference design for an authentication mechanism for nomadic environments. Using the framework, we have implemented a prototype authentication mechanism which is persistent and does not require user interaction. Additionally, it supports a user level delegation of authentication in a persistent manner. We refer to it as NDAF prototype, in the following text.

The prototype consists of several independent network applications and configuration files. The software part together with hardware, consisting of a desktop computer running Debian Linux and a RFID reader, form a complete NDAF prototype. Some of the software applications in the prototype are executed manually when required by a user, while others continue to run in background, as Linux daemons, until the computer is shutdown.

Following is the list of all modules which constitute NDAF prototype.

1. RFID-Reader: It is used to scan RFID tags, present in the interrogation field.
2. Computer running Debian Linux: Underlying hardware and the operating system.
3. Serial Port Library: It is used to communicate with RFID-Reader.
4. Authentication Server: It corresponds to NDAF Server and NDAF Fusion in the architecture of framework.
5. Authentication Client: It corresponds to NDAF clients of the framework.
6. GDMlogin: It is a classic password authentication program but with some custom modifications.
7. Configurator: It is an application, used to install the relevant files and authentication credentials.
8. DLG: It is a console application which is used to delegate or revoke authentication tokens.

The interaction of these modules in the prototype is shown in Figure 12. The authentication Server, which corresponds to both NDAF-Server and NDAF-Fusion of actual framework, is the heart of authentication mechanism implemented in NDAF prototype. It grants or denies an authentication request using the information which it receives from the RFID reader. As shown in red in the figure, Server is launched with the privileges of super user at startup, just before GNOME display manager (GDM) gets started. In GNOME based desktop, GDM manages sessions (X-Servers) of all users. Server in the prototype interacts with GDM, by sending commands in standard protocol format, through Clients which run in each of the user's sessions. GDM controls the computer display and provides an abstraction for multiple sessions present on a computer. GDM also invokes the password based login program (GDMlogin), whenever a new session is started or reactivated. However, GDMlogin program is only a front end for interacting with a user for the user name and the password. Actual authentication is achieved by a pluggable authentication module associated with GDM. This association is described in a special file in the Linux configuration folder.

After the start of Server at startup, it takes the control of RFID reader through the serial port library. It communicates with RFID Reader, Clients, GDMlogin and DLG applications to achieve persistent authentication and delegation. As shown in the figure, all these applications communicate with each other using network sockets. The use of network, for interprocess communication, enables us to port the authentication mechanism on an actual nomadic network where each terminal may run a remote session, similar to the concept in NDAF architecture described in the last chapter.

In typical use, a user carrying a valid RFID tag enters into the interrogation field of the RFID reader. The RFID reader reports its identification to 'Server', which checks the identification in the local data

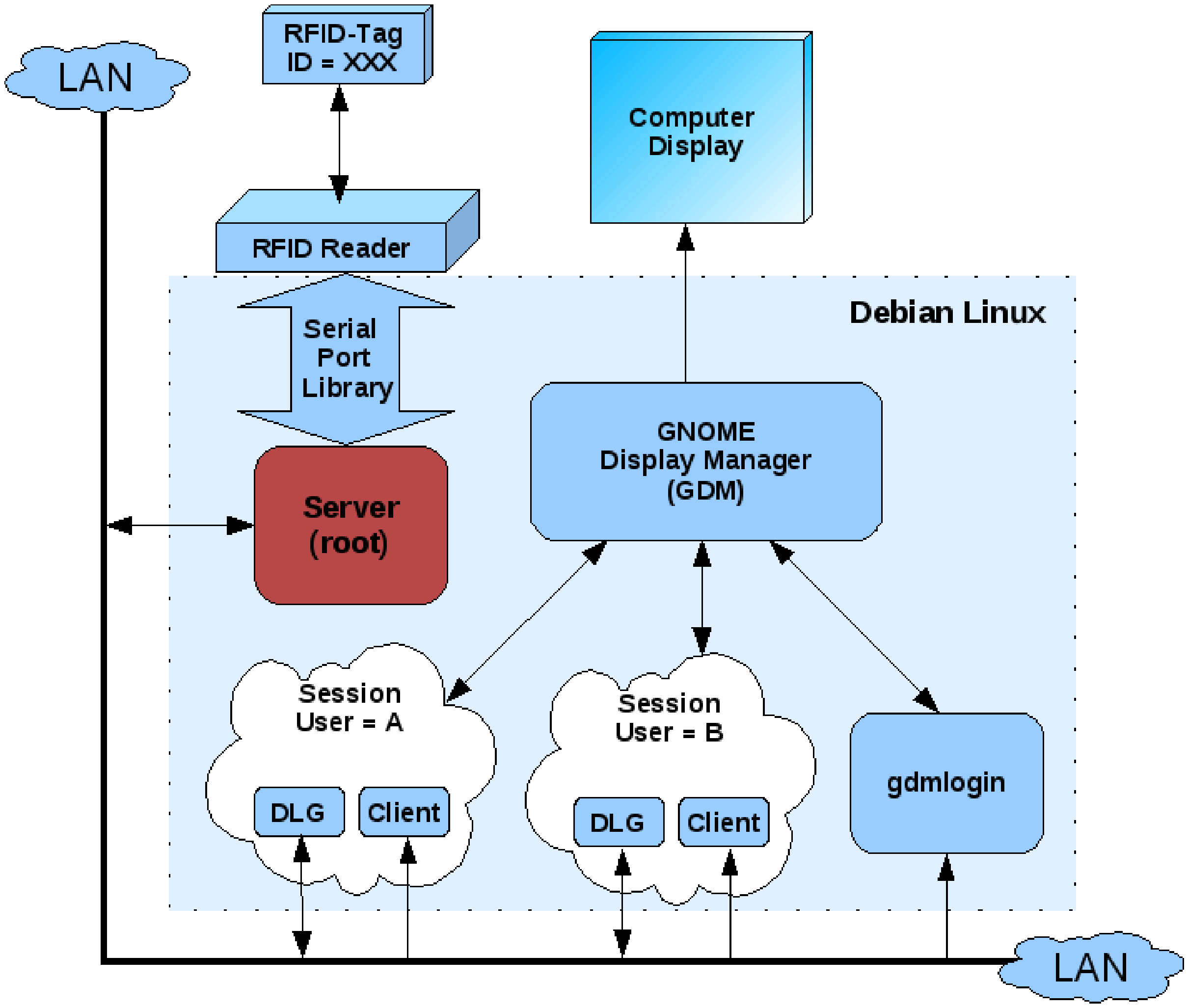

Figure 12: Interaction of Runtime Modules in NDAF

base to find a match for a system user. If a match is found then Server checks the availability of its session. If a relevant session is found, Server sends a command to the corresponding Client so that it can activate the session on the computer display. However, if a session is not found, Server creates a new session and waits for a predefined period of time to allow the new session to start. If a person is already authenticated and using his session then Server does not switch context even if another valid user enters the field of RFID reader. More detail on the individual modules is presented in the next section.

Design Units

In this section, we do not provide the internal detail and architecture of RFID reader[6], Debian Linux[54] and Serial Port Library[38], which can be found from the corresponding reference of original source. We consider them as block box modules for the prototype. The rest of the modules are described in detail.

1. RFID Reader

The main purpose of RFID reader hardware is to scan the limited field of space, also called interrogation range, and send the list of identifications of all RFID tags present in this field. Although, the reader can be programmed to perform more sophisticated jobs, we have used it only for reading identification codes. More detail on the working of RFID reader and its relevant application programming interface (API) can be found in Appendix A.

2. Debian Linux

In general, Debian refers to an operating system (OS) which is entirely composed of free and open source software. Debian Linux is popular for its availability for a large number of industry standard hardware platforms, such as Intel 32/64 bit, AMD 32/64 bit, ARM 16/32 bit, zSeries Mainframe, etc. Many popular distributions of operating systems are based on Debian Linux, such as UBUNTU, MEPIS, Dream Linux, etc. Its standard installation uses the GNOME desktop environment. More detail information on its installation, availability, programming and architecture can be found on the Debian homepage.[54]

3. Serial Port Library

The internal design detail of the library can be found from its source website[38]. Here, we describe only the interface necessary to integrate it with the authentication Server. The top level API of the serial port library is in form of following function.

```
void mainExt2(int Parm_Count, const char *Parms[],bool EnableExt,
                SerDrv2_Data *data)
```

where

Parm_Count = Number of parameters in the second argument 'Parms' of function. In our case, its value is seven.

Parms = It is a string, specifying the configuration parameters for serial communication.

EnableExt = It is is a binary flag, when true, invokes the custom behavior of library necessary while using with NDAF, otherwise the library works in interactive mode.

Data = This is a pointer to a special data structure, used for exchanging data between an application and library.

The serial port configuration parameters used in the NDAF are as follows;

```
char *Parms[] = {" ","/dev/ttyUSB0","115200","8","1","0","6"}
```

It implies that we are using USB to serial converter at port zero at 115200 bits per second, with 8 bit word length, one stop bit and no parity. The last parameter '6' specifies the library to use plain text format while returning received data from the serial port.

The format of data structure, used to exchange data, is as follows.

```
char *in, int inL
char *out, int outL
```

The pointer 'in' is the address of a buffer where received data is stored. The size of this received buffer is specified in the integer 'inL'. Similarly, 'out' is the address of transmit buffer and 'outL' is the total size of data that needs to be transmitted on the serial port.

4. Authentication 'Server'

The top level state machine, which represents the functionality of Server application, is shown in Figure 13 and it looks very symmetric. The authentication Server works in three states, which are Known, Unknown and New. These states correspond to three authentication phases: authenticated user, non-authentic user and the first time of authentic user. By an authentic user we refer to a user whose session is authorized to be invoked on the given terminal. State transition signals are generated periodically (by default after one second), to give persistence for authentication and delegation mechanisms. These signals are New-ID, Last-ID and Unknown-ID. The name of each state transition signal corresponds to the name

of state to which transition is targeted. For example, 'New-ID' always causes transition to 'New' state, 'Last-ID' always causes transition to 'Known' state and the signal 'Unknown-ID' causes transition to 'Unknown' state.

Followings is a brief description of each state. Documentation of source code is in form of detailed comments added to the source listing.

1. **State=Unknown:** This state represents that there is no valid RFID tag in the radio field of reader.

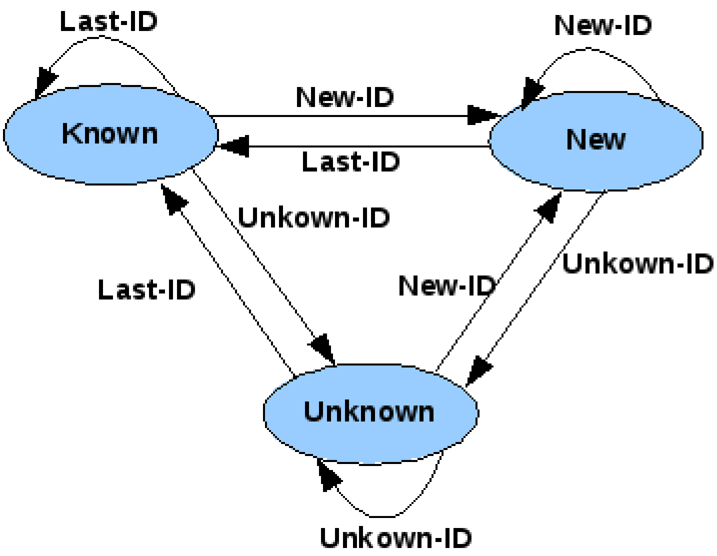

Figure 13: Authentication Server's State diagram

There could be a situation when some RFID tags are present in the field but their unique identification number does not match to any registered user in authentication mechanism's database. This state could also be entered in the case of poor reception for RF signals due to environmental radio noise. Since, the authentication Server is responsible for the critical security functions of authentication and delegation, thus on reaching this state, Server revokes all authentications and locks all sessions. Following functions are performed in this state.

   1. It broadcasts null string as the current user name, indicating to other modules the absence of users and causing them to fall back to respective fail-safe states.

   2. It decreases Confidence by one unit. Confidence is an integer variable and if greater than zero, it indicates that a valid user is present on the system for sufficiently long time. If Confidence becomes zero or negative, Server replaces RFID enabled pluggable authentication modules with default password based authentication. It also locks the console of currently active session with a password enabled screen saver.

2. **State=New:** This state is reached whenever a new and valid RFID tag enters in the radio field. A valid tag should be registered with the authentication mechanism. The tag being new implies that the user which corresponds to this tag is different from the user whose session is currently active or locked on the display. At this state, all functions, which are necessary to start a new session or re-activate an existing session, are performed. Some of them are as follows.

   1. The authentication Server broadcasts the user name which corresponds to the tag currently scanned.

   2. Server activates RFID enabled PAM module, so that user can be authenticated without entering his password.

3. If screen saver is active, it is unlocked.
4. Server sends a new session command to GNOME display manager (GDM) through a Client running in user space. As response, GDM starts a new session by launching an X server. The authentication Server waits for a predefined period of time to allow new session to start. It expects a feedback from the authentication Client running within new session. However, if it does not receive any reply, the new session is killed to avoid any security breach.
5. It switches the computer display to currently active virtual terminal, so that user can view his session.
6. It also locks or suspends all those sessions which do not belong to the current user.

3. **State=Known:** This state is reached whenever RFID reader reports the tag whose corresponding user is already authenticated. It means that same user is present across multiple authentication events, so it is a kind of steady state. In our prototype, the repeated authentication is necessary to achieve persistence. In this state, following functions are performed.
    1. It keeps the session unlocked.
    2. It switches the display to the virtual console terminal corresponding to the currently active user.

The authentication Server communicates using network sockets, with other modules of the authentication mechanism. The numerical values of sockets can be found in the header file. Here, we describe them by referring to their enumerated names. Following sockets are used for the interprocess communication.

- SESSION_FEEDBACK_SOCKET: This socket is used to receive messages from all Clients.
- USER_BROADCAST_SOCKET: It is used to broadcast user name in the system. Since, there are multiple Client applications in the system, therefore each Client is associated with a unique socket number in a set. This socket serves as base number for the set. For example the authentication Client corresponding to the first active session is USER_BROADCAST_SOCKET + 1, the second session Client is USER_BROADCAST_SOCKET + 2 and so on.
- USER_CMD_SOCKET: From this socket, Server receives commands from user programs. These commands are in a standard format. For example, DLG application sends delegation or revocation requests on this socket.
- USER_REPLY_SOCKET: This socket complements USER_CMD_SOCKET, and thus it is used to send replies to corresponding user program for their requests.

5. Authentication 'Client'

The authentication Client is a process which runs in user space with the corresponding user's privileges. This means, on a given system, there are multiple copies of Client application corresponding to each user. Server keeps track of all Clients running in a particular session and only sends the session specific commands.

Client serves two purposes.

1. It receives commands from Server and executes them in the current user space. However, it does not know which commands are being sent, as the authentication Client always receive a formatted string and execute it on console. In this sense, it may be considered as an access point for Server in a particular session.

2. It sends information about the user session to Server, by reading local environment variables such as current virtual terminal, user name, home directory, etc. Although, Server already has expected values, but this information provides more reliability, as there could be a case when the default behavior of Linux is not invoked, for instance fail-safe behavior due to some run time error.

6. GDMlogin

The purpose of Gdmlogin application is to interact with a user for obtaining the user-name and the password. By default this application is invoked by Gnome Display Manager (GDM) for authentication purpose. The Gdmlogin application does not actually verify users password but instead it forwards this information to the GDM, which invokes the appropriate Pluggable Authentication Module (PAM) for the verification. Thus Gdmlogin can be considered as the front end of password based login. The default interface is shown in Figure 14. This GUI can be configured with variety of different theme.

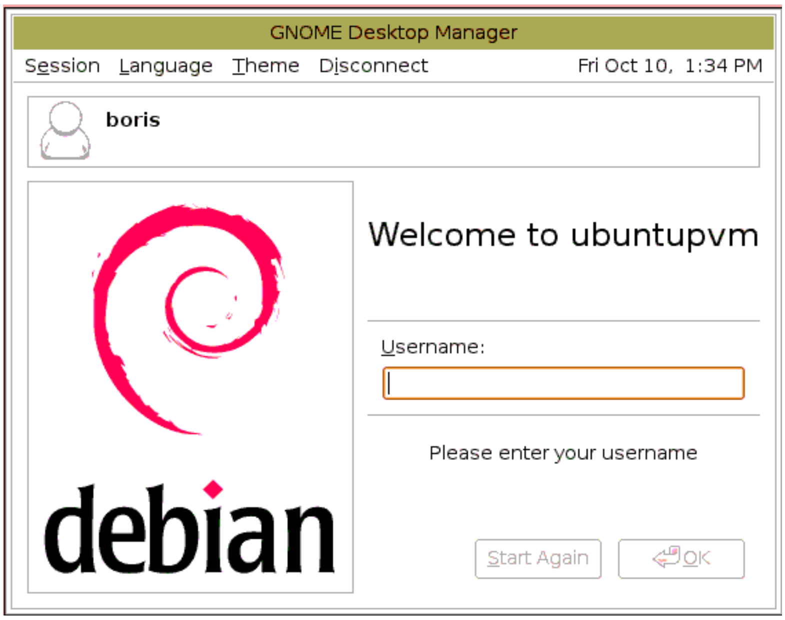

Figure 14: Gdmlogin Snapshot

We have modified the Gdmlogin application to include network support as it was indicated in Figure 12. With our custom Gdmlogin program, it is possible to get user-name and authentication credential from the authentication Server over a network.

7. Configurator

Although this module is not a part of normal authentication process, but it helps the system administrator to install required files and certificates. When a system is configured first time, the system administrator runs this application to register all users and their RFID tags. Similarly when new users need to be registered with the prototype, this application is run to update existing record. More information on its use can be found in Appendix B.

8. DLG

DLG is a command line application to control user's authentication level delegation. The implementations of delegation mechanism is also persistent which means that the delegation takes effect immediately soon after the command is issued. For example, as soon as a delegator revokes the

delegation, the delegatee immediately returns to his original session.

This application communicates with the authentication Server using USER_CMD_SOCKET by sending commands in a standard format. The standard format for Server is as follows:

<32 bit Operation Code><Sender name><Data String><8 bit Nonce>

A nonce is used to distinguish a fresh reply from Server from an old reply. Operation codes relevant to DLG application and corresponding data strings are as follows.

1. SET_DELEG: This code requests to set a delegation right from the user who sends this command to the delegatee whose name is present in data string of command. On command line delegation command is as follows; `'dlg set <delegatee>'.`
2. RESET_DELEG: This command is used to revoke a previously granted delegation. Delegation of the user who sends this command is canceled immediately by forcing delegatee to logout. On command line this operation is as follows; `'dlg reset <delegatee>'.`
3. RESET_REC_DELEG: This commands is used by a delegatee to cancel a delegation which he has received. The revocation by delegatee is required in many use cases. For example to alarm the delegator that delegatee is no more responsible for a particular job. The command line reference of this operation is; `'dlg reset-rec'.`
4. SWITCH_USER: This command is used by delegatee to switch to one of account whose delegation is previously granted. On command line this operation is accomplished by issuing following command; `'dlg switch <delegator>'`
5. GET_DELG: This could be most used command in delegation process, as it displays all inbound and outbound delegation. On command line it looks like; `'dlg get'.`
6. RESET_DELEG: This command cancel all outbound delegations instantly with out issuing individual commands for each user. Command line reference is as follows; `dlg reset-all'.`

More detail on implementation is in the form of commented source code. In the next chapter, we have described a detailed analysis of the prototype with respect to usability and security. We have also presented a performance comparison with password based authentication, in nomadic context.

CHAPTER - 6

Evaluation

As described in earlier chapters, most of the securely designed systems are not fully secure in practice and thus, there is a clear difference between designed security and effective security. In the case of nomadic users, this difference is quite significant due to special usability constraints imposed by their work environment. The purpose of the Nomadic Delegation and Authentication Framework (NDAF) and corresponding prototype is to minimize these type of usability constraints for authentication mechanisms and thereby increase the level of effective security. As we are interested in the usability perspective only, thus the use of specific access control mechanisms or underlying cryptographic primitives and protocols are beyond the scope of this research.

We have analyzed our system from two different angles. Firstly, from a usability perspective, we have investigated where our prototype stands, when compared to classic knowledge based authentication. This comparison is focused on those security vulnerabilities caused by usability requirements of nomadic users. Secondly, from pure security perspective, we have analyzed the prototype in order to find out design specific vulnerabilities. As it may happen that while addressing the vulnerabilities of knowledge based authentication, one may introduce new security vulnerabilities. At last, we present a performance comparison, to highlight a pure usability gain achieved by the prototype. This gain is in terms of time which a user saves, as a result of using the authentication mechanism of prototype. This gain also contributes to job satisfaction and increase in organizational productivity.

Usability impact

The usability of the authentication and delegation process greatly impacts the security, especially for nomadic users. As shown earlier, nomadic users tend to bypass authentication mechanisms if usability is poor. Therefore, for authentication purposes, the prototype system provides a true ubiquitous interface. A user simply walks up to the computer, uses it and then walks away. The authentication mechanism of our prototype takes care of the underlying authentication being used in the access control of the computer. Additionally, the mechanism supports authentication management for multiple sessions present on a single computer, which simplifies context based user switching.

For delegation purpose, active user involvement for explicit consent is necessary to avoid many security threats. This is because the consent for a delegation is saved in the system configuration until it is explicitly changed. It is different from an authentication event, where user consent may not be necessary if the mechanism supports persistent authentication, similar to our prototype implementation. The NDAF prototype uses a simple and single command to obtain the user consent for delegation or revocation. Similarly, a delegatee can access the delegator account by issuing a single command. From authentication point of view, this process is equivalent to sharing password with colleagues but is relatively more secure, because the delegator does not actually give away his authentication token. Moreover, the delegatee is accountable of all actions he performs, no matter that he acts as the delegator and possesses all privileges. This delegation process is also very simple compared to the delegation of roles or permissions in role based access control (RBAC) systems.

Now let us start by considering the security vulnerabilities associated with knowledge based authentication when used by nomadic users, which were also described in Chapter 3. We consider these vulnerabilities one by one and analyze them for our prototype. The analysis result is shown in Table 1.

S/N	Vulnerability	Threat in Knowledge based authentication	Relevancy to our authentication mechanism
1	Short or easy to type passwords	This may cause a dictionary based brute force attack.	This is not valid in the NDAF prototype, as a user does not use password in his normal work flow.
2	Forgetting passwords	This may cause a denial of service.	As the prototype uses device centric authentication, so the user does not interactively involve in the authentication process and thus there is no possibility for this vulnerability.
3	Stealing written long or complex passwords	Attacker may get unauthorized access.	This vulnerability is also not valid as a user does not have the knowledge of the secret used in authentications devices.
4	Forget to logout	Attacker may get a chance for unauthorized access by hijacking an already authenticated session.	The NDAF prototype supports persistence of authentication, which implies that a user is logged out as soon as he departs from the scene.
5	Password sniffing by an observer	Attacker may get unauthorized access. Even a partially observed password may be helpful in brute force attack.	As the prototype uses the wireless technology of RFID, sniffing is not possible by simple observation. However, sophisticated radio equipment can be used to listen communication, but it is still relatively more difficult than simple observations.
6	Giving away passwords for delegation purpose	By doing so, the authentication token is compromised and may get more compromised if the person is not trustworthy with whom password is shared.	The NDAF authentication mechanism supports a simple user level delegation, so a user does not necessarily need to hand over his authentication token. It minimizes the motivation for giving away an authentication token, by introducing a user friendly and easy delegation.
7	Sharing password in a group	The action compromises the authentication token beside there is no accountability while using authorizations.	With the user level delegation in the prototype, there is no particular motivation for users to cause this type of vulnerability. Besides, authentication tokens are RFID tags which may not be easily cloned.
8	Failure in revocation	This may cause unnoticed and unauthorized access for a period of time.	In the NDAF prototype, a user level delegation can be revoked instantly by issuing a single command by the delegator. Also a user can easily observe all active inbound and outbound delegations, which might be helpful if he forgets to revoke.

9	Non-persistent delegation	An attacker may get enough time for an unauthorized access, as revocation takes some time.	The NDAF prototype supports persistent delegation. This means as soon as delegation is revoked it takes effect immediately and the delegatee is not able to use the session even for a few seconds.
10	Frequent authentication	If the frequency of authentication is too high, it may cause denial of service as a user might be too busy in the authentication for the most of his time.	In the NDAF prototype, authentication process is automated and thus saves a user from wasting his precious time in the authentication process.

Table 1: Impact of Usability for Security

Thus, after analyzing these usability impacts of the NDAF prototype, it can be concluded that the developed prototype is able to remove a large set of security vulnerabilities which are caused by poor usability of knowledge based authentication. As those vulnerabilities are no more their, thus the effective level of system security is expected to increase.

Next is the analysis of our prototype from a pure security point of view. This is necessary to detect new vulnerabilities that might arise in the implementation process.

Security Analysis

For the purpose of security analysis, we use following symbolic annotations.

RFID Scanner = S

Authentication Server = A

User Client = U

Range of users = X

Range of RFID tags = ID

Range of system commands = Cmd

Range of system states = STATE (Known, New, Unknown)

$P \rightarrow Q$:Message = Source P has sent a message to the destination Q

P: <Function> = Entity P has performed a function

We have performed the security analysis for both authentication and delegation. Before going into detail, we present some underlying assumptions which form the basis for our security analysis of the NDAF prototype.

Underlying Assumptions

Some underlying assumptions related to system configuration are made for the purpose of this security analysis. The NDAF prototype is designed for Debian Linux and thus existence of standard Debian Linux is always assumed. The following Table 2 lists these assumptions along with their rationale.

S/N	Configuration Assumption	Rationale
1	There is only one super or administrative user.	This is necessary because our authentication Server is implemented with the assumption that no other system application can change its configuration at run time. Since, by default, the Debian Linux has only one root account so it is a quite reasonable and valid assumption, provided we do not explicitly make more accounts with root privileges.
2	The authentication Server A, is running as super user with administrative privileges and thus it can not be killed, stopped or modified by a user at run time.	The security of authentication decisions depends on the fact that the integrity of local database of a Server 'A' remains intact. This is a realistic assumption, because in the Debian Linux, no ordinary user can kill a process owned by the root user.
3	The authentication Server A, trusts RFID scanner S.	It is necessary as the protection of external hardware is not in the scope of this research. This is also a reasonable assumption provided following assumptions are also true. 1. An attacker does not have control over physical environment, e.g. attacker can not change the hardware, can not tamper with the computer, etc. 2. An attacker does not have the capability to install RF relay devices to mount distant relay attacks. Alternatively we can install RFID scanners with built-in capability to resist relay attacks, e.g. by measuring time to receive a response. 3. RFID scanner is working properly. We are not considering the possibility of faults that might occur in authentication related hardware.
4	Each user session has a valid copy of user's Client 'U', which continue to run as long as the session is running.	By default, each user session is configured to run the authentication Client U at start up of the relevant session. However, as this program is running with corresponding user privileges, a user may stop or tamper with it. However, if the user stops the authentication Client for extended period of time, the corresponding session is forced to terminate by Server A, as soon as it detects the absence of the U from the session. This is also quite reasonable assumption because a given user, even if he tamper with a Client, can not get authorizations to access the work space of another user or the root account.

5	The authentication Server 'A' trusts a user's Client 'U'.	This assumption implies that a user does not have the capability to tamper with the authentication Client U. Even if a user has the capability, then there is still a very little probability that he would actually use it. This is because, by doing so, he would not get any additional authorizations but in fact might make his own session less secure, which is least expected from a nomadic user. Moreover, we can always employ additional mechanisms to increase trust level if required, e.g, by periodically updating and restarting a Client 'U' or using encrypted communication.
6	Using network sockets for interprocess communication is secure.	This assumption is also necessary to narrow down the scope of our research as we are not looking for security issues related to network communication. In fact, our prototype uses plain text communication over UDP sockets, which is not realistic from security point of view. However, this choice is made to keep our focus on usability issues only. It is trivial to use the secure socket protocol (SSL[18] or TSL[19]) or encrypted communication and in that case this assumption holds well.

Table 2: Underlying Assumptions

Next we describe our analysis of the actual authentication mechanism implemented in the prototype. This analysis is based on above listed assumptions.

Analysis for the authentication

To analyze the authentication mechanism, we divide the authentication process in following seven steps. Possible vulnerabilities at each step are listed in Table 3.

Step	Description	Analysis
1	S → A: List of RFID tags (This step involves scanning the interrogation field of RFID reader S, and then sending the list of all detected tags to the authentication Server A.)	As per underlying third assumption, the list of scanned RFID should be valid and must corresponds to all tags present in the interrogation field. However, for S to trust this list, following conditions must also be true. 1. The tag is not cloned: If someone has made copies of passive tags, then the authentication mechanism produces false positive. This may be avoided by using tamper-resistant tags, e.g. zero-knowledge RFID as described by Engberg, Harning and Jensen[32]. 2. The tag still belongs to the owner and moreover, at the time of authentication, the owner is in the possession of this tag. This is an essential condition for any device centric authentication system and analysis for security vulnerabilities related to device centric approaches is beyond the scope.
2	A: ID = The preferred ID from the list of the first step (As the list of the first step may contains multiple identifications, it is necessary for Server to locate only one preferred identification. The preferred identification corresponds to the user currently login to the system.)	The security vulnerability which is possible in this case is the denial of service, due to the possibility of a user tag not being scanned. This might occur, for example, if there are a very large number of tags present in the field, the reader might miss certain tags during single scan. To mitigate its effect, the authentication Server does not switch context on the basis of single scan and instead works on the principle of moving average, where the rate of scanning is higher than the response time to initiate a context switch.
3	A: Map an ID to a user X (We use two types of authentication databases in our prototype. The first corresponds to all recognized RFID tags and the second corresponds to all valid user names present on the system. This mapping function represents all associations for a particular user towards multiple RFIDs. The mapping is implemented in the form of security certificates present in user's home directories.)	This involves finding the user-name that corresponds to a particular RFID and it depends on security certificates present in each user's home directory. All these certificates are signed to avoid tampering with and also a user does not have the permission for any modification. A user is not allowed to install a certificate that belongs to some other user, as the authentication Server A always performs search in the corresponding user's home directory.

4	A: process STATE and update internal data structures. (A particular state is reached based on the outcome of third step. A given state may requires some modification of internal data structures.)	Various implementation bugs or buffer overflow type of vulnerabilities may result in incorrect values to be stored in internal data structures, which might be used, in controlled manner, to launch some sophisticated attacks. We consider these type of vulnerabilities beyond the scope for this research. However, to avoid these, one may analyze the software and state machines by some formal verification method. Also, secure programming or security enhanced kernels might be helpful to combat these types of problems.
5	A → U : Cmd (In this step, the authentication Server A sends an authentication related command to a authentication Client U. These commands are used for locking, unlocking, activation and suspension of a particular user's session.)	As per underlying assumptions, the forth and the fifth, commands send by Server A are likely to be executed by Client U, with a high probability. But still, there exist some conditions, which might cause a failure. For instance, if there is a communication error while sending a command through a network socket, then this step might fail. However, dealing with these type errors caused by operating systems or communication networks are beyond the scope. If it really happens, following threats are possible. 1. Session may fail to lock and thus provides a chance to an attacker to access confidential information. 2. The display does not switch to the session corresponding to a currently authenticated user. 3. New session may not start, causing a denial of service. To avoid this, we may use some nonce based feedback channels, so that Server can retry or kill the session and force the system to fall back to a safe mode. Currently, this type of mechanism is only implemented for starting new sessions. If a new session fails to start the authentication Client within a time limit, Server kills all processes associated with the session.
6	A: STATE = Next STATE (The authentication Server calculates the next state of operation.)	If we assume hardware or communication failures beyond the scope, the only security vulnerability which we experience is an incorrect state transition when a user tag is not scanned by RFID reader, even if it is physically within the interrogation field. To avoid this problem, we have introduced the notion of Confidence in the authentication Server implementation. A context switch only takes place if the 'Confidence' drops below a certain threshold. Thus, our mechanism works on the moving average principle, where frequency of authentication is higher than the minimum response time for a context switch.

7	A: Go to the first step after 1s (At the end, the authentication Server goes to sleep for one second and then it jumps to the first step in the authentication mechanism.)	By default, the reader scans the interrogation field, once in each second, in order to avoid overloading the system. It also implies that the system response time is a couple of seconds. This might be a security vulnerability for automated authentication environments, as a given authentication is always few seconds old. However, the NDAF prototype is designed to authenticate humans, so few seconds response time is very reasonable when we compare it to a human response time. Thus, for this research we assume that a few seconds window is too short to be classified as a vulnerability.

Table 3: Step by Step Analysis for the Authentication Mechanism

The analysis for the authentication shows that the developed prototype is reasonably secure. Next, we start describing the analysis of delegation mechanism as implemented in the prototype.

Analysis for the delegation

We have analyzed delegation mechanism with help of a formal authentication logic[33], called BAN logic, named after its inventors, Burrows, Abadi and Needham. One may note that we have not used the BAN logic to prove our authentication logic. The primary reason is the complexity and numerous assumptions associated with our proposed Nomadic Delegation and Authentication Framework (NDAF), which do not allow us to perform such a detailed analysis in the limited time for this research. But the delegation support in the prototype is relatively simple and is quite suitable for the analysis using BAN logic.

Let us recall the sixth underlying assumption (i.e. secure interprocess communication) and for the sake of this analysis let us assume that it holds in the following form:

- All communication between the authentication Server A and the session Client U of user X is encrypted by a shared K_{XA}. This means that the key K_{XA} corresponds to ID of X and is different for each user.

Also, if a user X starts his session using K_{XA} then we may represent it by the following notion:

- $X \leftarrow K_{XA} \rightarrow A$

Now, we start applying the BAN logic. The first step in this regard is the conventional description of our delegation protocol, which is listed as follows:

Message 1: $X \rightarrow A$: $\{X,X',\text{'set'}\}K_{XA}$

Message 2: $X' \rightarrow A$: $\{X',X,\text{'switch'}\}K_{X'A}$

Message 3: $A \rightarrow U$: $\{Cmd\}K_{AX}$

The first message is a request, from a user X to the authentication Server A, to authorize the account delegation of the X to the user X'. This communication is protected by mutually shared key, due to the previously mentioned assumption of secure interprocess communication. In the second message the user X' requests Server A to switch him to the account of the user X. Again, due to the same argument used for the first message, this is also a protected communication. In the last message, Server A sends context switch command to the corresponding Client U, which enables the user X' to use the session of X.

The second step for the BAN logic is to translate the conventional description of the protocol to a form called idealized protocol form. Thus, the translation of the first step is as follows:

Translated Message 1: $X \rightarrow A$: $\{X \leftarrow K_{X'A} \rightarrow A\}K_{XA}$

We only include the first message, as this is sufficient to complete the delegation process. The second

and the third message merely activate the delegation and consequently do not contribute towards trust building.

Since, we assume a secure interprocess communication, thus we also consider following to be true:

1. X believes $X \leftarrow K_{XA} \rightarrow A$ (i.e. The delegator X believes in his secure communication with the authentication Server)
2. A believes $X \leftarrow K_{XA} \rightarrow A$ (i.e. The authentication Server believes in his secure communication with the delegator X)
3. X' believes $X' \leftarrow K_{X'A} \rightarrow A$ (i.e. The delegatee X' believes in his secure communication with the authentication Server)
4. A believes $X' \leftarrow K_{X'A} \rightarrow A$ (i.e. The authentication Server believes in his secure communication with the delegatee X')

Now, in the third step of BAN logic, we analyze the idealized form.

For the first message:

A sees $\{X \leftarrow K_{X'A} \rightarrow A\}K_{XA}$

since A believes $X \leftarrow K_{XA} \rightarrow A$, thus

A also believes X said $X \leftarrow K_{X'A} \rightarrow A$

and

A believes that X believes $X \leftarrow K_{X'A} \rightarrow A$

which concludes that A believes that the key $K_{X'A}$ of X' can be used to activate the session of the user X. In this way the trust build up completes, which is required for the delegation.

Performance Figures

As indicated earlier, poor usability lowers the effective security in nomadic environments. Thus, it is important to measure usability performance for the prototype. For this purpose, we have performed various performance trials using the physical setup depicted in Figure 15. The radius of RFID reader's interrogation field is 1.5m for the prototype. When a person enters in the RFID coverage area, his relevant session becomes unlocked and the session remains active as long as he is within the area. On his departure, the session becomes locked, once again.

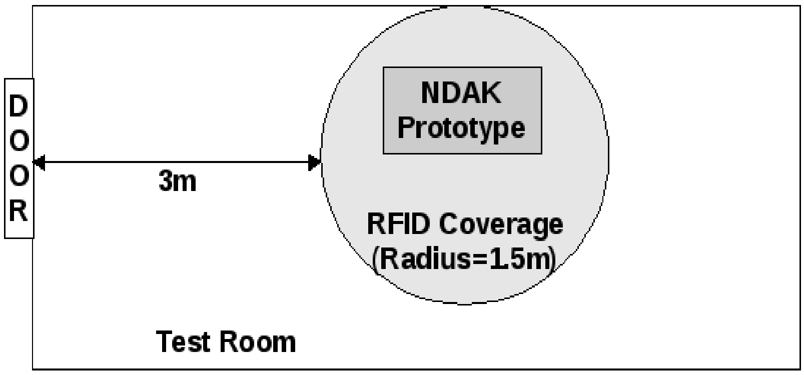

Figure 15: Setup for Performance Trials

We have measured the time taken during different phases of the authentication process, such as locking, unlocking, switching and starting of a new session, using both passwords based login and RFID token based authentication. Passwords used in these experiments are ten character long with at least two characters each from alphabets, numerics and special symbols. It is almost equivalent to 64 bit security. We have also gathered some measurements for the delegation process.

The following table summarizes the measurements, which might be subjective, as the setup does not fully reflect a true nomadic environment. Therefore, values differ depending on the physical layout of experiment and the type of users. All listed values are averaged and rounded to the nearest integer. For the comparison, we use a scale with range between -3 and 3. Positive values are used to indicated better performance of the prototype while negative values are used for the opposite case, i.e. password based login is better than the authentication mechanism of prototype.

S/N	Name of Performance Test	Measured Value		Comparison and Comments *0 = Equal,* *1 = better,* *2 = considerable gain,* *3 = considerable gain for both usability and security*
		NDAF Prototype	**Password based login (Length=10)**	
1	Session Lock Time	Automatic, Time ≃ 0 sec.	Manual, Time ≃ 1 sec.	+3 In the password based login system, a session may be locked using key board short cut or graphical user interface (GUI). Also there is a chance of forgetting to logout, which represents a security vulnerability. In the NDAF prototype, it is automated and does not provide any chance to an attacker for stealing the session.
2	Session Unlock Time	Automatic, Time ≃ 0 sec.	Manual, Time ≃ 8 sec.	+2 In the password based login, a user actively involves in the process which takes at least a few seconds. For nomadic users, this time is significant at the end of day, when we sum the time for all authentication events. Moreover, it motivates users to use short and easy to type passwords, which represents a security vulnerability.
3	Switching User	Automatic, Time ≃ 0 sec.	Manual, Time ≃ 10/25 sec. (25 seconds in the case of a system which does not support simultaneous sessions.)	+3 The user switching, in password based systems, involves at least two steps. Firstly, the current user must logout or lock his session. Secondly, the new user authenticates and starts the session. This process is quite poor from a usability point of view in a password based login. The NDAF prototype switches between users automatically and instantly.
4	Granting a Session Delegation	Time ≃ 2 seconds	Difficult to measure	+3 There is no good way of delegating a complete running session in the password based login for a Debian Linux system. The one possibility, although non-secure, is giving away one's password Delegating at permission or role level might be tedious and may also not produce desired results. Using 'sudo' commands for delegation process is not user friendly. On the other hand, for our prototype, it is just a matter of issuing a simple command.

5	Revoking a Session Delegation	Time ≃ 2 seconds	Difficult to measure	+3 This is not applicable for a password based login, as delegation is not possible in the first place. If a session delegation is achieved by the non-secure way using password sharing, then the only possible way for the revocation is to change the password. Obviously, this type of set up brings various security vulnerabilities as there is no accountability. However, in the case of NDAF prototype, a user revoke his delegation by issuing a simple command.
6	Switching to a delegated account	Time ≃ 2 seconds	Difficult to measure	+1 In the classic mechanism, if we assume that a user has successfully delegated all required permissions and roles then the delegatee does not need any switching. He simply uses delegated authorizations from his own session.
7	System Resources	~500MHz of Intel Celeron-D processor	Almost zero	-2 Since, a password based authentication mechanism does not supports persistent authentication and delegation, it does not consume any system resources at run time. While considering rapid advancement in computing technologies, this factor may become less significant in future.
8	New Session Creation Time	Automatic, Time ≃ 10 seconds	Manual, Time ≃ 20 seconds	+1 In classic systems, a user authenticates interactively before the system starts creating his new session. In the case of NDAF prototype, it is an automated and user friendly process.

Table 4: Performance Comparison

If we assume that, for a typical nomadic environment, on average a user authenticates every five minutes, then the number of authentication events in 8 hour work day sum to 96 events. According to the performance figures, the cost of 96 session switching is 960 seconds (~16 minutes per user per day) for a password based login and almost zero for the NDAF prototype. This difference gets larger if we also consider the possibility of delegation. Thus, we can speculate that the NDAF prototype provides an increase of about four percent, in the productivity of organization besides offering a good usability which may contribute to increasing job satisfaction.

CHPATER - 7

Final Words

We, as computer users, have evolved in the past four decades of computing history. This evolution is characterized by the way we fulfill our computing needs. In the start, we used to physically go to computing devices located at fixed locations, in order to meet our computing requirements. Over the period of time, we start using mobile computing devices which can be carried around where ever we move in an environment. More recently, we have started using computing devices that are embedded in a environment at all those places where one might need them. Thus, we no longer need to physically go to fixed locations for computing, nor do we need to carry around personal computing devices. We refer to it as nomadic use of computing.

Unfortunately, the evolution to nomadic use of computing is not properly accounted for with use of appropriate security mechanisms. The most obvious example of this is an password based authentication mechanism, which is a prerequisite to various access control mechanisms found in popular operating systems such as Unix, Linux, Windows and Mac OS. As nomadic users frequently authenticate while moving in the environment, the password based authentication presents poor usability due to its basic requirement of conscious human interaction. Due to usability constraints, the productivity of such nomadic environments is reduced and moreover, nomadic users tend to circumvent the authentication mechanism, causing numerous security vulnerabilities.

In this research, we have analyzed the interaction of nomadic users and knowledge based authentication mechanisms. As a result, we have found a number of security vulnerabilities resulting from usability constraints present in classic knowledge based authentication mechanisms. To address these security vulnerabilities, we have formulated a list of usability requirements for authentication mechanisms when used in a nomadic environment. Mostly, these requirements are based on a simple assumption that usability of security should be a part of security specifications. These usability requirements have helped us to design an authentication framework, called Nomadic Delegation and Authentication Framework (NDAF). We have evaluated NDAF by developing a prototype and perform usability and security analysis. Although the prototype implementation is a scaled down version of real life nomadic environments, it is sufficient for the proof of concept, and thus substantiate our hypothesis of increasing effective security by improving usability.

There are two types of gain which we achieve as a result of this research. The first is the improvement in the quality of interaction between humans and computers. This can be apprehended in terms of time, which our authentication mechanism saves by avoiding conscious human interaction, typically required by knowledge based authentication. The saved time contribute towards organizational productivity by efficient use of human resources beside providing more job satisfaction for the users themselves. For example, if NDAF based authentication mechanism is implemented in hospitals, we expect that doctors would get more time for patients which they currently consume while dealing with authentication mechanisms.

The second gain, which is the primary purpose of our research, is the improvement of security for a computer system. This security gain is attributed by the increase in level of effective security for authentication mechanisms in nomadic context. The expected increase is also shown in Figure 16, which is adapted from Figure 2 in the first chapter, to highlight the contribution of this research. As shown in the figure, a security mechanism is designed to meet certain security requirements, also known as the designed level of security. But, when it is deployed and practically used, the actual security it offers is usually less than the designed level, we refer to this reduced level as the effective level of security. Moreover, the effective security tends to decrease with the passage of time. There are various reasons for the difference between designed and effective level, but we mitigate one of them, known as the usability of security, with our prototype. Thus, we claim that this research helps to bring the effective level of security closer to the designed level. Our claim is supported by the security analysis which shows that we

indeed have removed many security vulnerabilities which are normally present in the interaction between a nomadic user and knowledge based authentication.

Although our research helps to decrease the security gap, we have no quantitative values for parameters or the exact shapes of curves shown in the figure below. This is because, the actual values of parameters vary with the type of a system, the characteristics of users, underlying cryptographic primitives, protocols, etc. Therefore, it is very hard to estimate them, but trends indicated in the graph are certainly true due to empirical observations and analysis presented in the report.

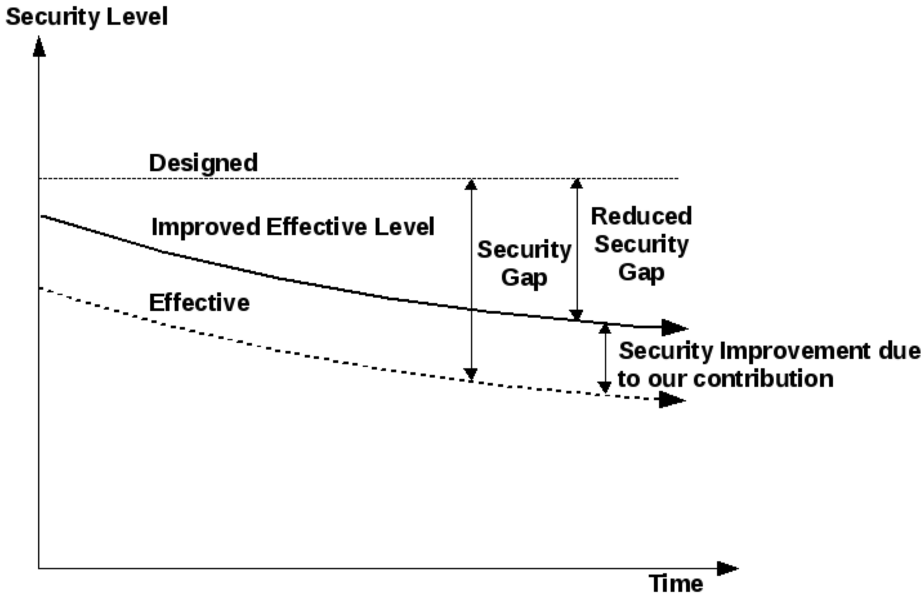

Figure 16: Security Impact of the Research

The importance of computer security has increased manyfold in recent years. We are much more dependent on machines for computation and information than ever before. Our desire to secure computing machines pushes security administrators to employ more sophisticated security mechanisms. At times, security personnel assess the security of a system independent of its usability. However, this approach is not always suitable. In some cases, security is highly dependent on its usability. The actual effect of usability varies across different systems and may depend on users, work environments, interaction patterns, available options for an interaction, etc. In such systems, usability should be an integral part of the security specifications otherwise the designed level of security may not be achieved in practical use. Even the best security mechanism, if it contains usability constraints, is not suitable and secure for all scenarios.

It is not always possible to achieve good usability and high security at the same time. That is why, security experts try to find out a good trade off between the two, under limited computational resources. Unfortunately, a good balance between usability and security is usually specific to a particular system under consideration. For instance, a password based authentication may be considered a very good balance of usability, security and system resources, for personal computing. But, as our research indicates, it is not true for nomadic computing where it offers poor usability and low security. Thus, it is necessary that security experts realize these conclusions and should always carefully consider the usability of security in their design decisions.

Limitations

The list of usability requirements, presented in this report, may not be sufficient and therefore many more requirements might be discovered by a much deeper analysis. Moreover, we have focused on knowledge based authentication, especially password based login mechanism, to compile the list of requirements. Apparently, knowledge based authentication is among the worst case from a usability point of view, but still a broader analysis might be more useful. At this stage, we also confess that a true analysis of a nomadic environment requires cross-disciplinary approach including anthropology and psychology, along with a couple of clinical and field trials, in order to truly discover the best usability options that may be incorporated in a security mechanism.

The proposed authentication framework may not be equally suitable for all nomadic environments. Some nomadic environments are much more distributed and it might be difficult to synchronize Clients and the authentication Server for a real time operation. Thus, a direct adoption of the framework to real environments may be non-trivial in some cases. Similarly, some nomadic environments may require distributed architecture for the authentication Server and Fusion, in order to meet certain dependability requirements.

The prototype implementation of the framework is a scaled down version and does not demonstrate the functionality of some modules such as context monitor, fusion, session migration, etc. In the prototype, we have used just one branch of multi factored authentication. We have not experimented with more than four users due to limitation of the single computer. Similarly, passive RFID tags have certain security vulnerabilities, which in some cases make them even worse than passwords. Therefore, it is necessary that for any real world application, secure, active and multiple tags should be used. Moreover, we have used plain text socket communication, which must be replaced by some secure method such as secure socket protocol[18][19]. The software implementation of the prototype might have some bugs, which must be fixed by some formal verification method. In spite of many drawbacks of the prototype, it serves its intended purpose and demonstrates usability aspects of authentication very well. But, before using it in real world, the security aspect of authentication mechanism can not be ignored.

Future Work

Further work may be continued in a number of directions. Some of them are indicated here.

1. A more complete and rich implementation of multi factor authentication can be experimented. This work may go beyond the framework which we have presented in the report. Experimenting with video tracking for context monitoring and using biometrics authentication techniques may be non-trivial in a network. Moreover, access control mechanisms can be modified to cope with probabilistic nature of authentication in multi factor algorithms. Under these circumstances, a complex authorization management may be required.

2. The NDAF prototype can be expanded to experiment on real network environment by invoking remote sessions. In this way, session migration techniques may be experimented to give more realistic user experience for nomadic environments.

3. A better user interface for delegation can be designed which might require no user interaction at all. This is quite possible in multi factor authentication, where an authentication mechanism is tracking the presence of multiple authentication tokens simultaneously. However, this is not a trivial task due to numerous associated security vulnerabilities. In this case, a more comprehensive security and usability analysis might be required. This problem is a good example of the situation where usability and security are horn locked.

4. The software implementation can be modeled in a more formal way to analyze any potential design problems in order to improve reliability and dependability. This may also help to uncover some of the implementation bugs. A true secure system is always thoroughly analyzed from both

theoretical design perspective and actual implementation perspective. This may lead designers to choose some other software language where formal verification is relatively easier.

5. The back-end of proposed Nomadic Delegation and Authentication Framework (NDAF) is a server that controls all sessions. This might be considered as a single point of failure. One can design the back-end on distributed systems and use some secure protocols for their real time synchronization.

6. The NDAF can be instantiated to a range of platforms other than the Debian Linux, such as Security Enhanced Linux, Windows, Free BSD and Apple's Macintosh operating system.

Appendices

Appendix A: Radio Frequency Identification

The material in this appendix is a summary of the documentation about RFID technology and the relevant product which we have used in our prototype NDAF, and can be located on Alien Technology home page[6]. For more detail on individual topics visit documentation section and data sheets of relevant RFID products on the home page.

Introduction

Radio-frequency identification (RFID) technology uses radio frequency signals to acquire data remotely from RFID tags that are present within interrogation range. The data can be used for variety of purpose, as in our case it is used for authentication purpose. The interrogation range of a RFID reader is typically few meters and may depend on some programmable options related to antenna signal strength.

A RFID system may consists of following components as shown in the figure;

1. RFID Reader (Interrogator): A reader may consist of following sub modules.
    1. Transmitter
    2. Receiver
    3. Microprocessor
    4. Antenna(s)
2. RFID Tags
3. Output device(s) and/or host computing device e.g. a personal computer.

A reader may be referred to as an "interrogator" because it interrogates tags for data or to perform certain actions. Anti-collision is the ability of a reader to read multiple tags in the field[6]. When reader send RF signals, all tags respond to this signal simultaneously. If reader does not have anti-collision capability, it would get confused and may not be able to read correct data.

A typical structure of a RFID system is shown in Figure 18. Host system communicates with microprocessor present on RFID reader in order to program it or send some instantaneous commands. Depending on programming options and commands received from host system, microprocessor activate transmitter and receiver to send and receive RF signals. These signals are actually transmitted by antenna whose transmit signal strength may determine the interrogation range. RFID tag present in the interrogation range respond to sent RF signals and method they used for this purpose depends on the type of tags, as discussed in following sections.

The antenna broadcasts the RF signals that are generated inside the reader's transmitter . In addition to this, the antenna also receives responses from tags within range. In general, readers may use more than one antennas to detect and interrogate , although typically at any given time only one antenna is active. Polarization often implies a preferred tag-antenna orientation which optimizes the antenna's ability to acquire tag signal and data[6].

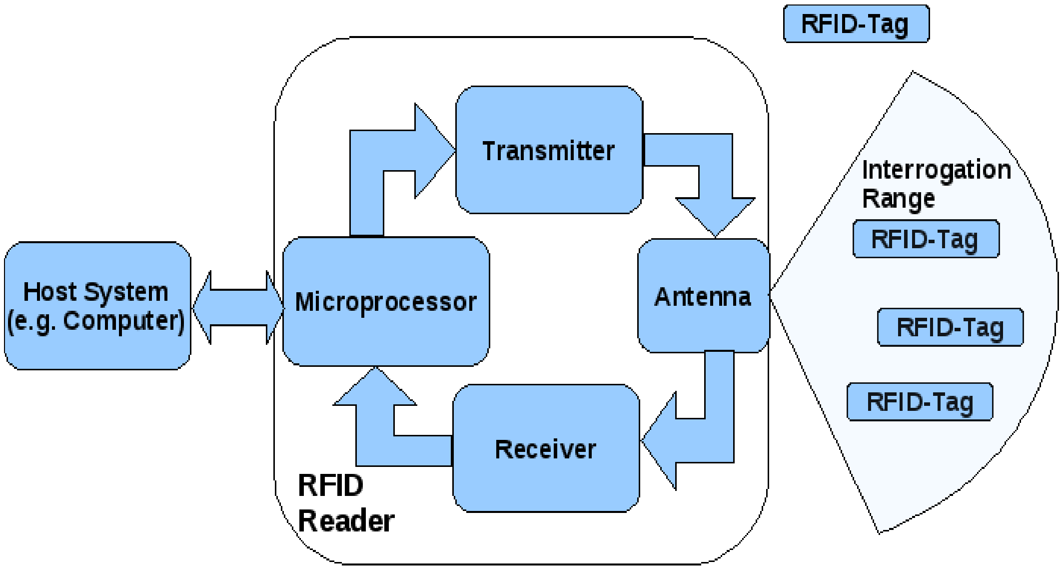

Figure 17: Typical RFID System

RFID Tags

RFID tags are micro devices that contain identification and other information that can be communicated to a reader from a distance. However, RFID tags can contain much more data and can be read at greater distances. Some RFID tags can be read under more challenging wireless conditions, and in some cases can accept new data.

Tags are often classified as either passive or active to describe how they communicate with the reader. As name indicates, passive means the tag uses a modified form of the reader's own signal to send back its data. On other hand, active means the tag contains its own transmitter. The term active and passive is also used in other context as to distinguish read only tags or tags containing microprocessor. However, in our discussion we restrict ourself only to former description of active and passive tags.

Passive Tags

A passive tag, in a communications context, relies solely on backscatter modulation of a reader's signal for communication with the reader. A passive tag in this context is one that has no transmitter of its own[6]. A passive tag, in a power perspective, derives its power exclusively from the energy contained in the reader's incident signal. Similarly, a passive tag in information processing context, has no on board processor but it is a relatively obscure use of term[6]. In this report, we use the term passive tag to refer its communication context only.

A passive tag uses a method called "modulated backscatter" to convey its data to the reader. As shown in Figure 18, the tag reflects (or backscatters) the RF signal transmitted by the reader and embeds its unique ID and data by modulating that reflected signal. Th backscatter is a modulation technique in which communication from tag to reader uses radio frequency (RF) energy broadcast from a reader/antenna. Thus it essentially bounce the RF signal off the tag and back to the reader[6]. A backscatter tag is physically incapable of communicating data outside the presence of a reader's interrogation range. As shown in following figure, reader and tag have a unique identification. In normal usage reader asks some question and tag respond to this question using backscatter method. This means that a tag is only able to respond to reader's questions and it can not transmit data of its own.

Tag range depends on many other factors besides the characteristics of the tag. A RFID reader's transmitting power and receiver sensitivity, reader's antenna range and relative polarization, and the reading environment can all affect the range at which a given tag may be successfully read. Usually, tags affect system performance and should be configured to optimize for the specific applications under consideration.

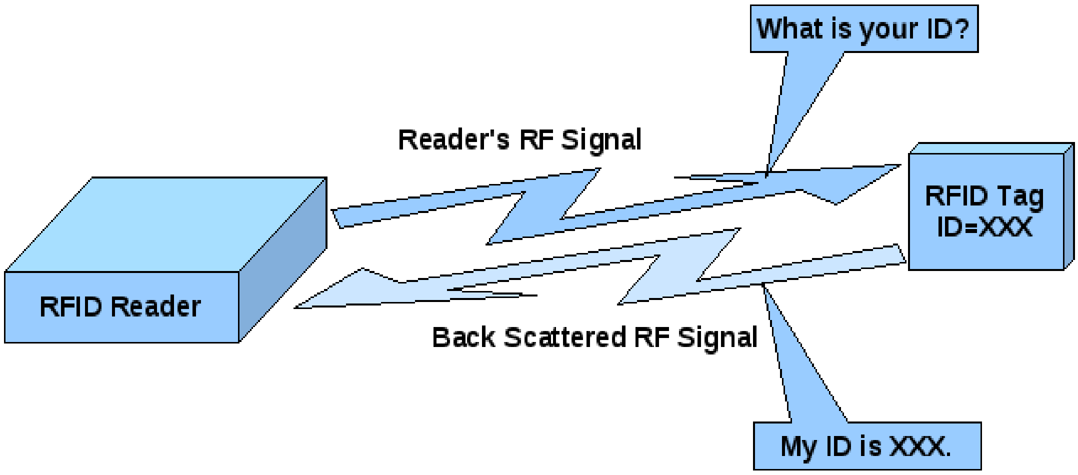

Figure 18: Backscatter Mechanism for Passive RFID Tags

Tags come with different memory types. Some tags may have just enough memory to hold only the simplest of information such as 64 bit identification. Some may have as much memory and processing power as a small computer. Tag memory may be read-only, write-once-read- many (WORM) or read/write (R/W).

Active Tags

From a communication perspective, a tag that contains its own transmitter, which can broadcast data even when no reader is present is called an active tag. Usually, such a tag derives its power exclusively from a source other than the reader’s radio signals. This source could be a battery, cell, power adapter, etc. Sometimes, an active tag contains an on board microprocessor. This type of tag can process data which it receives or stored in local memory[6]. Active tags may be considered to be either transmitters or transponders, though, to be precise, a transponder is always a transmitter tag, but not all transmitter tags are transponders. A transmitter tag can broadcast a message into the environment even if there is no reader active. But each transponder tag only transmits in response to the reader’s command.

RFID tags are also divided in classes, depending on their resources. A 'Class-1 Tag' is ultra-low-cost, beam-powered backscatter tag which contains up to 96 bits of data. It usually has write once read only memory (WORM) storage, no on board processor and optimal read range of about one meter[6]. A 'Class 2 Tag' is low-cost, beam-powered backscatter tag with read/write memory, no on-board processor and optimal read range of 1 meter[6]. Similarly, a 'Class 3 Tag' is medium-cost, battery-powered backscatter tag with read/write memory, an on board processor and optimal read range of up to 10 meters[6].

Here, we have not discussed the active tags technology in detail, as we have used only passive tags in our implementation.

Host Computer

For any practical use, the RFID system needs either a host computer to process that data or some kind of output function that responds to the tag data. In most of the cases, both host computer and output functions are used.

With help of a host computer, the RFID system can log and process tag transactions for desired purpose. For example it can be used encrypt/decrypt file system or set authentication permission in response to presence of specific tags. Similarly host may involve in more complex operation like periodically updating tag data in order to achieve security and privacy.

We can program microprocessor of RFID reader to perform certain output functions. The simplest RFID system may only react with specified outputs according to a set of rules programmed into the

reader's microprocessor. For example we can program that if a certain tag is detected, output on a particular pin should go high, which in turns can be used for various other purposes like switching lights on and off, unlocking a car etc.

An RFID system may also be designed to react to certain input signals. For example readers are often configured to interface with presence detectors. A presence detector can also be used, to power up a reader only when an object is within range in order to save energy or to minimize the radio noise.

Alien RFID System (ALR-8780)

We have used Alien Technology's ALR-8780 RFID system in our prototype. This section is essentially a summary of relevant ALR-8780 documentation[6] customized for the project in order to make reproduction of the prototype as easy as possible.

The equipment used is an Agile frequency transmitter operating in a frequency range of 865.6 MHz to 867.6 MHz over 10 channels spaced 200 kHz apart. Agile frequency transmitter allows to use single antenna for multiple transmitter operating at different frequency. The equipment is used to interrogate passive RFID tags which work on the principle of radar backscatter and are powered from the RF energy emitted from the reader.

Some other specifications of ALR-8780 system are as follows.

1. Type of Modulation: AM (amplitude modulation)
2. Number of Channels: Ten (10)
3. Transmit Ratio: 4 seconds on, 100ms off
4. (LBT) Listen Before Talk: Active on all channels
5. Type of Antennas: Patch type antennas with 6 dB gain and 69 degree beam width
6. RF Power from Reader Ports: Less than 30dBm
7. Antenna Management: Time Division Multiplexed (TDM)
8. Transmit Power from Antenna: Less than 2 watt - ERP (Effective Radiated Power)

Class 1 NanoBlock Tags

The ALR-8780 RFID readers are designed to read and program 'NanoBlock' Tag, which are Alien's Class-1 128-bit tags. These tags comply with the EPC (Electronic Product Code) specification. Class 1 tags are "passive" devices, which means that they do not have an on board power source and are powered by the RF energy transmitted by the reader.

Alien's Class I tags contain 96 bits of programmable memory,out of which 64 bits are programmable by user. The remaining bits are controlled by the reader to keep track of state and checksum information.

Hardware setup

A proper hardware setup is required for the smooth working of authentication mechanism Following instructions should be followed carefully. Reader antennas should be positioned so that personnel in the area may safely remain at least 25.5 cm from the antenna's surface. Antennas should be mounted at the periphery of the desired read window (either overhead or side-wise). Since we have used two antennas in the prototype, so one should mount the second antenna in a mirror- image of the first antenna's position. RFID Reader should be positioned close to the antenna and the listen antenna must be mounted in a vertical orientation.

The host computer should be in a position that person sitting in front of terminal is in the range of

RFID reader. This makes sure that person may be successfully authenticated during repeated scans. The coaxial cables should be routed from the antennas to the reader and must be secured properly. The power supply connector should be pushed into the reader port and one must not forget to tighten the retaining ring finger. The female end of the power cord must be plugged into the power supply and the male end of the power cord into the mains outlet. One should align the RS-232 connector with the corresponding serial port on the reader and must push the connector onto the pins. One must finger-tight the screws to secure the cable to the reader.

After power on, we should wait for complete boot up. At this stage hardware setup is complete and now we can test it through some serial terminal communication program.

Host-Reader Interface of Alien's ALR-8780

This section is a summary of complete Reader Interface Guide of ALR0780/ALR8780 and can be found on Alien Technology website[6]. However, in this section, we present customized configuration which are necessary for proper working of the prototype.

Host-Reader interface supports both Ethernet and serial communication but command set is same for both cases. In our prototype we have connected through serial port. Connecting through Ethernet is a trivial task which only requires an additional Ethernet port on the host computer.

One should connect crossed null modem cable between host and ALR-8780. One may use USB to RS232 converter if a host computer does not have RS232 port. We also have used USB to RS232 converter in the prototype. ALR-8780 works only with following configurations: 115200 bps, eight bit data, no parity, one stop bit and no flow control.

During normal operation, the reader always has the list of active tags that are present in the radio field. Active tags are those which are read by the reader at least once within a predefined interval. New tags presented to the reader are added to the list, and tags that have not been seen for a period of time (Persist Time) are removed from the list. At any time, host system can interrogate reader to retrieve this list of tags. Autonomous Mode is a configuration and operation mode that enables automated monitoring and handling of tag data. While reading tags in Interactive Mode is a matter of issuing a single command to the reader. This command is `“get TagList”`. The result of the command is a list of tags that the reader can see. We use interactive mode of operation in NDAF prototype, as we have experienced its more reliability as compared to autonomous mode.

Currently, the prototype does not configure RFID hardware automatically. A user must configure ALR-8780 hardware with sequence of commands that are listed below. The user does not need to repeat this process after each power up provided he saves the configuration in the reader's flash. For configuring first time, a user must repeat following sequence of commands.

1) get Function / set Function

These commands are used to get or set configuration about current mode of operation. The Reader could be operated in two modes of operation. One is for programming, i.e. writing data to tags, while other mode is reading, i.e. reading data from tags.

For proper operation, each RFID tag registered in NDAF prototype must have a unique identification. To write a custom identification number to tag following commands must be issued.

```
set Function = Programmer

program tag = <64 bit hexadecimal custom identification number>
```

For returning to normal NDAF authentication mechanism, following command must be issued;

```
set Function = Reader
```

2) get TagList

This command gets the current list of active tags from the reader. It is called by the prototype periodically to poll the list of available tags.

3) get PersistTime / set PersistTime

These commands get or set the persistence time for tags in the tag list. Persist times are specified in seconds. A zero persist time guarantees that tags are not stored in the tag list while setting the persist time to -1 causes the history to build indefinitely. For the prototype following command must be issued to set persist time of two seconds.

```
Set PersistTime = 2
```

4) get TagListFormat / set TagListFormat

These commands set or get the format for tag lists. Following formats are supported in reader.

1. Text: Tag lists displayed as plain text messages, one tag ID per line.
2. Terse: Tag lists displayed as plain text messages, one tag ID per line, but just ID, count, and antenna, with no labels.
3. XML: Tag lists are displayed in XML text format.
4. Custom: Tag lists are displayed in the format described by TagListCustomFormat.

For the prototype following command must be issued to set TagListFormat to Custom.

```
Set TagListFormat = Custom
```

5) get TagListCustomFormat / set TagListCustomFormat

These commands specify custom formats for text based custom tag lists. For the prototype following command must be issued for proper operation.

```
set TagListCustomFormat = #%i
```

It sets the format in such a way that output, for four tags in the field of interrogation, looks like as follows.

```
#<RFID-1>#<RFID-2>#<RFID-3>#<RFID-4>#
```

6) get AutoMode / set AutoMode

These commands switch auto mode On or Off. As we are not using autonomous mode, following command must be issued.

```
set AutoMode = Off
```

7) get NotifyMode /set NotifyMode

These commands switch notify mode on and off. As we are not using auto notification, following command must be issued.

```
set NotifyMode = Off
```

8) Save

This command saves current settings to the flash memory of RFID reader. If configuration is saved then we do not need to issue same sequence of commands at each boot up. So after configuring reader with help of all commands described in this section, save command should be issued.

Appendix B: NDAF Prototype: User Manual

The prototype of Nomadic Delegation and Authentication Framework (NDAF) is a collection of applications and configuration files, developed for a Debian Linux based system. Since, the prototype uses GNOME desktop thus it also uses the GNOME Display Manager (GDM) and the GDM login program. The common example of this configuration is UBUNTU, which is the most popular Linux distributions in the computer world[43].

S/N	File Name	Default Location	Function
1	Authentication Server	/usr/local/bin	It automatically runs with the privileges of a root user and it corresponds to two modules in classic NDAF architecture, which are Server module and Fusion module.
2	Authentication Client	/usr/local/bin	On the prototype, multiple copies of this application are run automatically, with user level privileges. Each copy corresponds to a particular session currently present on the prototype
3	Configurator	Installation Directory	This application needs to be run manually and with the privileges of root user. It may be be considered as setup program which configures the system by installing all required files and security certificates.
4	Server.sh	/etc/init.d	It is a script which is automatically invoked at Linux run level 2, just before the GDM get started.
5	DLG	/etc/local/bin	This is a console application and it is invoked by a user when he needs to perform some function related to delegation.
6	GDM (PAM)	/etc/pam.d	It is a configuration file of pluggable authentication module (PAM) for Gnome Display Manager (GDM). It contains information about underlying authentication modules to be used with the GDM authentication process.
7	LibNDAF_PAM.so	/lib/security	It is a shared library for PAM and contains custom authentication routines. These routines are called when an application that is configured with this library requests for authentication. For our case, it is GDM that requests authentication functions in this library.
8	Setup.sh	Installation Directory	This script invokes the Configurator with the root privileges.
9	Client Configuration	~/.config/autostart	It is a configuration file whose purpose is to start a Client at the start of each new session.

Table 5: List of Files for the NDAF Prototype

The list of files for NDAF prototype, along with their default locations and brief functions, is shown in

Table 5. Some applications on the prototype run automatically while other need to be run manually

It is necessary to configure a system before actually using NDAF based authentication. After the successful completion of configuration, only thing user has to do is to carry a valid RFID authentication tag. The rest of detail of the functionality related to the authentication and delegation is automated.

Configuring a system

In the installation folder present on the accompanied disk, all necessary files listed in Table 5 are present. For installation of these files to their default location, system administrator should run setup.sh script. It asks for the root password in a console window because installation can not be completed with out root level privileges After successfully entering the password, a startup window similar to Figure 19 appears.

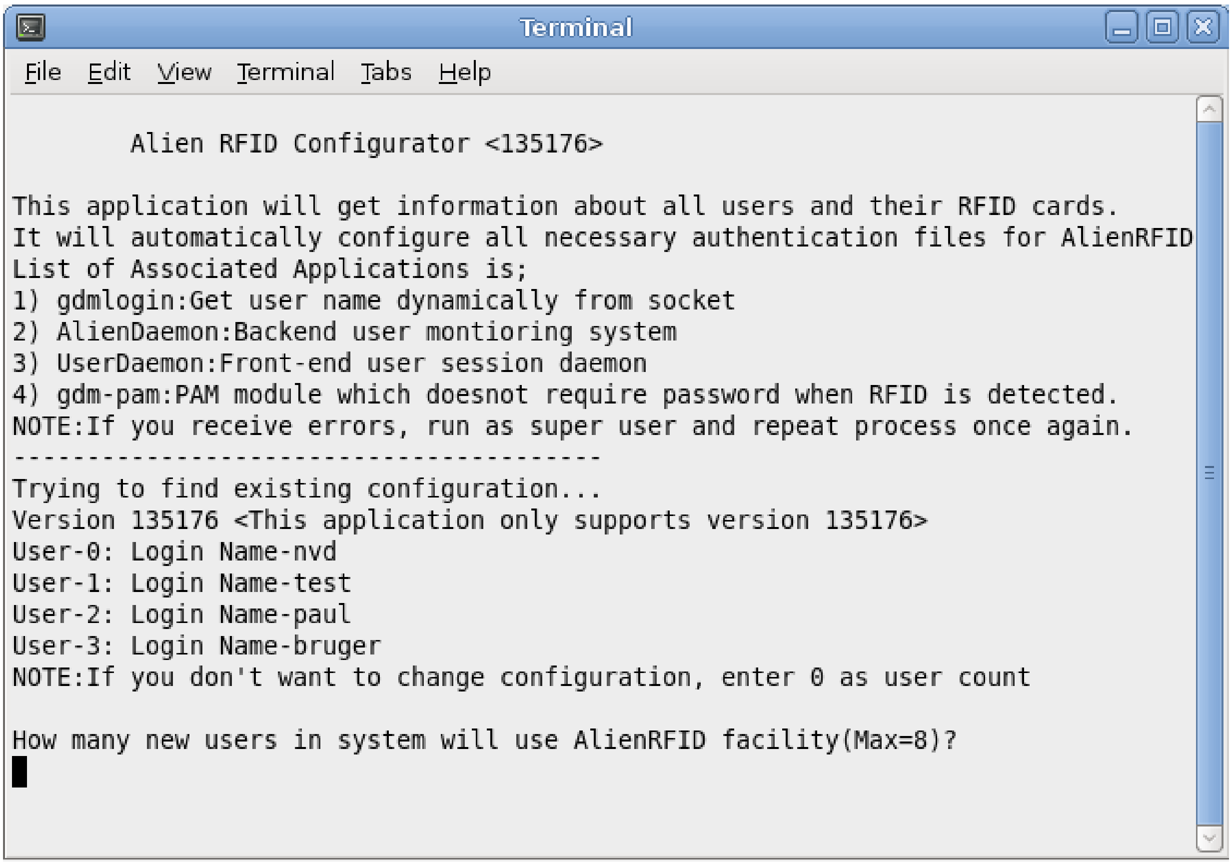

Figure 19: Configurator Screen shot

This startup window presents an important but implementation dependent information, which should be considered before proceeding further. In the first line there is a configuration identifier (in the figure it is 135176). It corresponds to the format of supported configuration of the application. As Configurator application is not backward compatible, it can not be used to update the configuration database which had been written in some old format. In the second half of the startup window, a version number of existing configuration is written. If one wants to update the existing configuration, this version number should be same as the configuration identifier present in the first line.

The lower half of the start up window also lists all users who are currently registered on the system. At this stage, Configurator asks for number of users that one wants to add. Further, it also asks for the choice, for either appending new users or replacing existing users. In any case, Configurator then starts asking user name and requests for placing their RFID tag in the field. This process continues until all users are registered on the system. After configuration is complete a system restart is required to enable

these updates.

Using the system

The NDAF based system is characterized by its simplicity and seamless interface and we can use it for both authentication and delegation. For authentication, a user does not have to do anything special except to carry the RFID tag which he has registered with NDAF in the configuration process. After this, user simply walks up to the system and when he is in the range of RFID reader, his session is automatically activated on the computer. Similarly, when user walks away and come out of the interrogation field of RFID reader, his session is locked.

For the delegation purpose 'dlg' command is used along with an action string. To illustrate some simple delegations, let us assume that there are two users registered on the system, named Alice and Bob.

To view current status of all inbound and outbound delegations, use following command.

```
> dlg get
```

If Alice wants to delegate her session to Bob, she issues following command.

```
>dlg set Bob
```

Similarly, if Bob wants to switch to the session of Alice, he uses following command from his own session.

```
>dlg switch Alice
```

Finally, if Alice wants to revoke the delegation which she has granted to Bob, following command is used.

```
>dlg reset bob
```

Appendix C: Source Code

Source code and executable files can be found on the accompanied disk.

References

[1] Min Li Hua Wang, “ABDM: An extended flexible delegation model in RBAC”, 8th IEEE International Conference on Computer and Information Technology, CIT 2008, Volume 8 , Issue 11, Pages 390–395, July 2008.

[2] Jason Crampton and Hemanth Khambhammettu, “Delegation in role-based access control”, International Journal of Information Security, Volume 7, Issue 2, Pages 123-136, April 2008.

[3] Hua Wang and Jinli Cao, “Delegation, Revocation and Authorization”, Lecture Notes in Computer Science 4928, Book; Business Process Management Workshops, Pages 294–305, Springer-Verlag Berlin Heidelberg, 2008.

[4] David Ferraiolo, Rick Kuhn and Ravi Sandhu, “RBAC Standard Rationale: Comments on A Critique of the ANSI Standard on Role-Based Access Control”, IEEE Security and Privacy Archive, Volume 5 , Issue 6, ISSN:1540-7993, November 2007.

[5] Jacques Wainer and Akhil Kumar, “A Fine-grained, Controllable, User-to-User Delegation Method in RBAC”, Proceedings of the 10th ACM Symposium on Access Control Models and Technologies, Pages 59–66, ISBN:1-59593-045-0, 2005.

[6] Alien Technology Corporation, 18220 Butterfield Blvd., Morgan Hill, CA 9503, USA, Homepage <http://www.alientechnology.com>, Last Visited March 23rd, 2009.

[7] Mark D. Corner and Brian D. Noble, “Zero-interaction authentication”, Proceedings of the 8th Annual International Conference on Mobile Computing and Networking Atlanta, Georgia, USA, Pages 1– 11, ISBN:1-58113-486-X, 2002.

[8] NIST Role Based Access Control (RBAC) and Role Based Security, <http://csrc.nist.gov/groups/SNS/ rbac>, Last Visited March 24th , 2009.

[9] David F. Ferraiolo and D. Richard Kuhn, “Role-based Access Control”, 15th National Computer Security Conference Baltimore MD , Page 554 – 563 , 1992.

[10] Ravi Sandhu, David Ferraiolo and Rick Kuhn, “NIST Model of role based access control”, Information Technology Lab, National Institute of Standards and Technology, <www.itl.nist.gov>, 2000.

[11] Jakob E. Bardram, Rasmus E. Kjær, and Michael Ø. Pedersen, “Context-Aware User Authentication– Supporting Proximity-Based Login in Pervasive”, Book; UbiComp 2003: Ubiquitous Computing, ISBN 978-3-540-20301-8, Pages 107-123, Springer Link, October 30, 2003.

[12] Mark D. Corner, Brian D. Noble, “Protecting applications with transient authentication”, International Conference On Mobile Systems, Proceedings of the 1st international conference on Mobile systems San Francisco, California, Pages: 57 – 70, 2003.

[13] F. Bennett, T. Richardson, and A. Harter, “Teleporting- Making Applications Mobile”, Proceedings of the IEEE Workshop on Mobile Computer Systems and Applications, Pages 82–84, ISBN 978-0-7695-3451-0 , IEEE Computer Society USA,1994.

[14] B. Brumitt, B. Meyers, J. Krumm, A. Kern and S. Shafer, “EasyLiving: Technologies for Intelligent Environments”, Springer Berlin/Heidelberg ISSN 0302-9743 (Print), 1611-3349 (Online), Pages 12-29, Volume 1927/2000, January 01, 2000.

[15] A. Ward, A. Jones, and A. Hopper, “A New Location Technique for the Active Office”, IEEE Personal Communications, Volume 4, Issue 5, Pages 42-47, ISSN: 1070-9916, October 1997.

[16] Daniel M. Russell and Rich Gossweiler, “On the Design of Personal & Communal Large Information Scale Appliances”, Book; UbiComp 2001: Ubiquitous Computing, ISBN 978-3-540-42614-1, Pages 354-361, January 01, 2001.

[17] Ensure Technologies, Xyloc Family of Products, Homepage <http://www.ensuretech.com>, Last Visited March 24th, 2009.

[18] Alan O. Freier, Philip Karlton, Paul C. Kocher, "The SSL Protocol Version 3.0", Transport Layer Security Working Group, 1996.

[19] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", Network Working Group, Request for Comments: 5246 , August 2008.

[20] Kris Nagel, Cory D.Kidd, O'Connell, Thomas O'Connell, Anind Dey and Gregory D. Abowd, "The Family Intercom: Developing a Context-Aware Audio Communication System", Proceedings of UbiComp 2001, DOI 10.1007/3-540-45427-6, Pages 176-183, 2001.

[21] R. Want, A. Hopper, V. Falco, and J. Gibbons, "The Active Badge Location System," ACM Transaction on Information Systems, Volume 10, Issue 1, Pages 91-102, January 1992.

[22] Jonathan Trevor, David M. Hilbert and Bill N. Schilit, "Issues in Personalizing Shared Ubiquitous Devices", Proceedings of the 4th International Conference on Ubiquitous Computing, Göteborg, Sweden, Lecture Notes In Computer Science, Volume 2498, Pages 56 – 72, ISBN:3-540-44267-7, 2002.

[23] Trusted Computer System Evaluation Criteria of National Institute of Standards and Technology (NIST) dated 1985, <http://csrc.nist.gov/publications/history/dod85.pdf>, Last Visited March 30th, 2009.

[24] Lawrence A. Tomei , "Encyclopedia of Information Technology Curriculum Integration", Information Science Reference; Illustrated edition , ISBN-13: 978-1599048819, February 5th, 2008.

[25] Mike Ebbers, Wayne O'Brien and Bill Ogden, "Introduction to the New Mainframe: z/OS Basics" dated July 2006, <http://publibz.boulder.ibm.com/zoslib/pdf/zosbasic.pdf>, Last Visited March 26th, 2009.

[26] Ezedin S. Barka - Framework for Role-Based Delegation Models, A Dissertation Submitted to the Graduate Faculty of George Mason University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy Information Technology , 2002.

[27] Pam Snaith and Rob Steiskal, "Mainframes are Still Mainstream", White paper by CA Inc. dated June 2007, Homepage <www.ca.com>, Last visited March 30th, 2009.

[28] Mark Weasor, "Nomadic Issues in Ubiquitous Computing", Xerox PARC, <http://www.ubiq.com/ hypertext/ weiser/NomadicInteractive> , Last Visited March 26th, 2009.

[29] Marcia Riley, "Ubiquitous Computing: An Interesting New Paradigm", <http://www.cc.gatech.edu/classes/cs6751_97_fall/projects/say-cheese/marcia/mfinal.html>, Last Visited March 26th, 2009.

[30] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege and D. Spence, "Network Working Group: Request for Comments No. 2904" dated August 2000, <http://www.ietf.org/rfc/rfc2904.txt>, Last Visited March 30th, 2009.

[31] Brandon James Carroll, "Cisco Access Control Security: AAA Administration Services", Cisco Press. ISBN-13: 978-1-58705-124-1, May 27th, 2004.

[32] Stephan J. Engberg, Morten B. Harning and Christian Damsgaard Jensen, "Zero-knowledge Device Authentication: Privacy & Security Enhanced RFID preserving Business Value and Consumer Convenience", In Proceedings of the 2nd Annual Conference on Privacy, Security and Trust (PST'04), 2004.

[33] Michael Burrows, Martin Abadi and Roger Needham, "A logic of authentication", ACM Transactions on Computer Systems (TOCS), Volume 8 , Issue 1, Pages 18 - 36 , ISSN 0734-2071, February 1990.

[34] Jakob Bardram, Thomas Kjær and Christina Nielsen, "Mobility in Health care - Reporting on our initial Observations and Pilot Study", Technical Report of a Clinical Study, CfPC 2003-PB-52, Center for Pervasive Computing, 2003.

[35] Jens Bæk Jørgensen and Claus Bossen, "Executable Use Cases for Pervasive Health care", IEEE Software

Volume 21 , Issue 2, Pages 34 – 41, ISSN 0740-7459, March 2004.

[36] Jakob Bardram, “The Trouble with Login: on Usability and Computer Security in Ubiquitous Computing”, Personal and Ubiquitous Computing, Volume 9 , Issue 6, Pages 357–367, ISSN 1617-4909, November 2005.

[37] Rachna Dhamija and Adrian Perrig, “Deja Vu: A User Study Using Images for Authentication”, In the Proceedings of the 9th USENIX Security Symposium, Denver, Colorado, August 2000.

[38] The computer technology documentation project: A Linux serial port test program, <http://www.comptechdoc.org/os/linux/programming/c/linux_pgcserial.html>, Last Visited April 1st, 2009.

[39] Matt Bishop, Book “Computer Security: Art and Science” , Addison-Wesley Professional, ISBN-13: 978-201440997, 2002.

[40] Computer Industry Almanac, “25-Year PC Anniversary Statistics”, Press release August-2006, <http://www.c-i-a.com/pr0806.htm>, Last Visited April 1st, 2009.

[41] Password Research, “Authentication Statistic Index” maintained by Bruce K. Marshall, <http://passwordresearch.com/stats/statindex.html>, Last Visited April 1st, 2009.

[42] Science Daily: Science News, "Age-related Memory Loss Tied To Slip In Filtering Information Quickly” dated 5 September 2008, University of California, San Francisco, CA 94143, <http://www.sciencedaily.com/releases/2008/09/080902143234.htm#>, Last Visited April 1st, 2009.

[43] Ladislav Bodnar, “Top Ten Linux Distributions”, <http://distrowatch.com/>, Last Visited April 1st, 2009.

[44] Lawrence O’Gorman, “Comparing Passwords, Tokens, and Biometrics for User Authentication”, Proceedings of the IEEE Volume 91, Issue 12, Pages 2019-2040, 2003.

[45] Charles P. Pfleeger and Shari Lawrence Pfleeger, “Security in Computing”, Prentice Hall Professional Technical Reference, 2002.

[46] Daniel V. Klein, “Foiling the cracker: A Survey of, and Improvements to, Password Security”, In Proceedings of the Second USENIX Workshop on Security, Pages 5-14, July 1990.

[47] Bruce L. Riddle, Murray S. Miron, and Judith A. Semo, “Passwords in use in a university timesharing environment”, Computers and Security, Volume 8 , Issue 7, Pages 569–578, ISSN 0167-4048, November 1989.

[48] Einar Jonsson, “Co-Authentication - A Probabilistic Approach to Authentication”, Masters thesis, IMM-Thesis-2007-83, Department of Informatics and Mathematical Modeling, Technical University of Denmark (DTU), 2007.

[49] Federal Communication Commission (FCC), “Performance and Accountability Report” for period October 1st, 2007 to September 30th, 2008, <http://www.fcc.gov/Reports/ar2008.pdf>, Last Visited April 7th, 2009.

[50] Cambridge Encyclopedia, Homepage <http://encyclopedia.stateuniversity.com/>, Last Visited 15th, April, 2009.

[51] Robert W. Proctor, Kim-Phuong L. Vu, “Stimulus-Response Compatibility Principles: Data, Theory, and Application”, Book; ISBN-13: 978-0415315364, March 24, 2006.

[52] Open Wall Project, Homepage <http://www.openwall.com/> , Last Visited April 15th, 2009.

[53] Martin Kirschmeyer, Mads S. Hansen and Christian D. Jensen, “Persistent Authentication in Smart Environments”, 2nd International Workshop on Combining Context with Trust, Security and Privacy, Trondheim, Norway, 2008.

[54] Debian Linux, Software in the Public Interest (SPI) Inc. , Homepage <http://www.debian.org/>, Last Visited April 15th, 2009.

[55] Butler W. Lampson, “Protection”, ACM SIGOPS Operating Systems Review, Volume 8 , Issue 1, Pages 18-24, Xerox Corporation, Palo Alto, California, 1974.